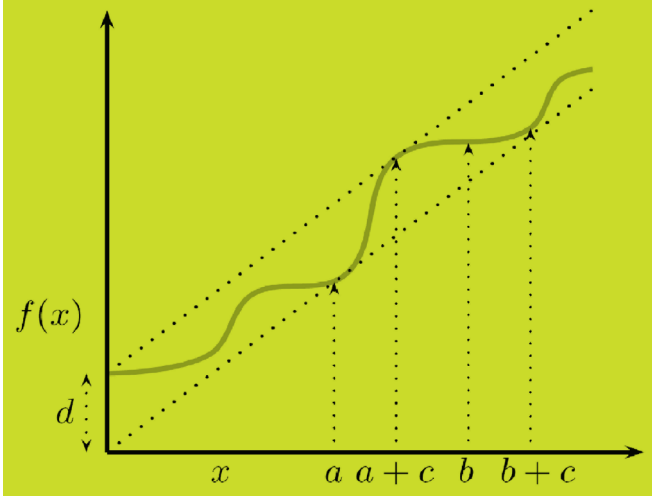
Routing and scheduling problems

Reinhardt, Line Blander; Pisinger, David; Madsen, Oli B.G.; Kallehauge, Brian

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Routing and scheduling problems



PhD thesis 15.2011

DTU Management Engineering

Line Blander Reinhardt
June 2011

DTU Management Engineering
Department of Management Engineering

Ph.D. thesis

Routing and scheduling problems

Line Blander Reinhardt

May, 2011

Dansk titel:
Ruteplanlægning
Type: Ph.d.-afhandling

Forfatter: Line Blander Reinhardt

ISBN-nr:

DTU Management
Department of Management Engineering
Technical University of Denmark

Produktionstorvet, Building 426,
DK-2800 Kgs. Lyngby, Denmark
Phone: +45 45 25 48 00, Fax: +45 45 25 48 05
phd@man.dtu.dk
May, 2011

Summary

In today's globalized society, transport contributes to our daily life in many different ways. The production of the parts for a shelf ready product may take place on several continents and our travel between home and work, vacation travel and business trips has increased in distance the last couple of decades. To deliver competitive service and price, transportation today needs to be cost effective. A company requiring for things to be shipped will aim at having the freight shipped as cheaply as possible while often satisfying certain time constraints. For the transportation company, the effectiveness of the network is of importance aiming at satisfying as many costumer demands as possible at a low cost. Routing represent a path between locations such as an origin and destination for the object routed. Sometimes routing has a time dimension as well as the physical paths. This may be that the objects routed have an availability time window and a delivery time window or that locations on the path have a service time window.

When routing moving transportation objects such as vehicles and vessels schedules are made in connection with the routing. Such schedules represent the time for the presence of a connection between two locations. This could be an urban bus schedule where busses are routed and this routing creates a bus schedule which the passengers between locations use.

In this thesis various routing and scheduling problems will be presented. The topics covered will be routing from an origin to a destination on a predefined network, the routing and scheduling of vessels in a liner shipping network given a demand forecast to be covered, the routing of manpower and vehicles transporting disabled passengers in an airport and the vehicle routing with time windows where one version studied includes edge set cost making the cost of the individual vehicle routes inter-dependant.

Depending on the problem type, the size of the problems and time available for solving, different solution methods can be applicable. In this thesis both heuristic methods and several exact methods are investigated depending on the problems needed to be solved.

The solution methods applied to the problems cover dynamic programming for multi constrained shortest paths, Branch-and-cut for liner shipping, Simulated annealing for transporting assisted passengers in airports, branch-cut-and-price for vehicle routing with time windows and edges set costs.

Dansk Resume

I det aktuelle globaliserede samfund bidrager transport til vores daglige liv på mange forskellige måder. Produktionen af dele i et hylde færdigt produkt kan foregå på flere kontinenter og afstandene på rejser mellem hjem og arbejde samt forretnings og ferie rejser er blevet længere de seneste årtier. For at kunne levere en konkurrence dygtig pris samt service niveau har transport udbyderen i dag behov for at yde god service med en lav omkostning. En virksomhed, der har enheder eller dele af enheder som skal fragtes, vil stile mod at have en billig fragtbefordring, mens visse tidsmæssige betingelser ofte skal være opfyldt.

Forskellige rutnings problemer bliver præsenteret i denne afhandling. De præsenterede emner er rutning mellem to lokaliteter på et allerede fastlagt netværk, rutning og skedulering af container skibe for container skibsfarts netværk hvor forespørgsler skal i møde kommes, rutning af arbejdere i en lufthavn som assistere passagerer med reduceret mobilitet og rutning af køretøjer med tidsvindues betingelser hvor den præsenterede version inkluderer kant set omkostning som gør prisen for de enkelte køretøjers ruter indbyrdes afhængige.

Når man planlægger ruter for transport objekter såsom køretøjer og fartøjer, så bliver der ofte lavet en tidsplan for afgange og ankomster på ruterne. Disse tidsplaner repræsenterer indirekte et tidspunkt for en forbindelse mellem to lokaliteter. Eksempler på dette er bus køreplaner hvor busruterne skaber en bus køreplan som passagerne kan bruge til at planlægge deres rejse.

I denne afhandling er der blevet formuleret problemer som har udspring fra situationer i det virkelige liv. Disse er løst med forskellige løsningsmetoder alt afhængig af problem type, størrelsen af problemet og tiden til rådighed for at finde en løsning.

De anvendte løsningsmetoder er dynamisk programmering som er brugt til korteste veje med flere objekt funktioner, Branch-and-cut som er brugt til containerskibsfart, simuleret udglødning som er brugt til transport af assisterede passagerer i lufthavne og branch-cut-and-price, som er brugt til et sær tilfælde af køretøjs rutnings problemer med tidsvinduer med omkostning for adgang til forskellige sæt af kanter.

Preface

This thesis was prepared in part at DTU Transport and in part at DTU Management, at the Technical University of Denmark as part of fulfilling the requirements for acquiring the Ph.D. degree in engineering. The work presented was supervised by Professor David Pisinger, Professor Oli B. G. Madsen and Assistant Professor Brian Kallehauge.

This thesis deals with modeling and solving different problems concerning routing and scheduling.

The thesis contains a general overview of the routing and scheduling problems and a collection of 6 research papers prepared during the period from August 2007 to April 2011.

Lyngby, May 2011

Line Blander Reinhardt

Scientific papers included in this thesis

- **Chapter 4:** Line Blander Reinhardt and David Pisinger. Multi-Objective and Multi-Constrained Non-Additive Shortest Path Problems. *Computers and Operations Research*, 2011.
- **Chapter 5:** Line Blander Reinhardt and David Pisinger. A Branch and Cut algorithm for the container shipping network design problem. (Conditionally accepted).
- **Chapter 6:** Line Blander Reinhardt, Tommy Clausen and David Pisinger. Dial-a-ride with synchronization for handicap assistance at airports. (Submitted). Technical report 2010.
- **Chapter 7:** Line Blander Reinhardt, Mads Kehlet Jepsen and David Pisinger. The vehicle routing problem with edge set costs. (Submitted). Technical report 2011.
- **Chapter 8:** Kenneth Hvam, Line Blander Reinhardt, Pawel Winter and Martin Zachariasen Bounding component sizes of two-connected Steiner networks. *Information Processing Letters*, 2007. vol 104.
- **Chapter 9:** Line Blander Reinhardt, David Pisinger and Amelia Regan. Root Balanced Minimum Spanning Graph: Algorithm and Complexity. Working paper.

Acknowledgements

I would like to thank my supervisor Professor David Pisinger for his guidance and for believing in me. I am thankful to Brian Kallehauge and Oli B.G. Madsen for supporting the PhD project and Brian Kallehauge for supervising me through my first year of study. I thank David Pisinger for not hesitating to take me on as a PhD student and for always making time for me when needed. I also want to thank colleagues at DTU Transport and DTU Management for creating a wonderful work environment.

I wish to thank Professor Amelia Regan and Professor Sandy Irani at University of California Irvine. Amelia Regan for inviting me and supporting me during my stay and for being there even during her time of grief. Sandy Irani for our inspirational meetings at UCI.

Special thanks goes to all my co-authors and reviewers for their cooperation and their contributions to the research in the individual projects: Tommy Clausen, Mads Jepsen, Amelia Regan, Kenneth Hvam, Pawel Winter, Martin Zachariasen and David Pisinger with whom it was a pleasure to work with as co-authors. Jonas Villumsen, Tor Justesen, Patrick Lincoln, Julia Lawall, Amelia Regan, David Pisinger, Christian Edinger Munk Plum, Berit Løfstedt, Shahin Gelareh, Jose Fernando Alvarez, Charlotte Vilhelmsen, Brian Kallehauge, Torben Barth, Richard Lusby and Simon Spoorendonk for reviewing parts of this thesis.

Last but not least, I would like to thank my husband for his support and for being a super husband and dad. I want to thank my children and my husband for not complaining about my absence and for their supporting hugs and kisses.

Contents

Part I

Synopsis

Chapter 1

Introduction

The globalization and the associated transport is apparent in our every day life when we look at the bottom of a coffee cup and it says "Made in China". Many of the products we use every day are or contain parts which are manufactured on different continents. Also the passenger transport service is, especially in Europe, part of our daily life when our children takes the bus to school and we go to work by public transport or our grandparents take a wheelchair taxi to their activities. Even when accessing the internet and when planning out heating lines or optic fiber lines routing occurs. To ensure cheap prices of products and good mobility of the workforce it is important to ensure a cost efficient transport with the desired service level. This can be done by planning the routes of the transporting objects well. Such route planning can also involve reducing the CO_2 emission to comply with regulations and achieving an environmentally friendly image for a company.

A large area in operations research involves routing and scheduling. This field involves everything from routing public transport over routing containers to routing optic fibers for data transport. Routing can be defined as selecting a path of travel. Operations research is often applied in real-life to generate valid and profitable routes. In the last decade, mathematical modeling has been applied to real-life routing problems and large gains are reported using these systems, see e.g. [7] for Swedish home care and [10] for Missouri lottery distribution. This thesis covers a variety of different routing problems which all have non-standard constraints inspired by real-life problems. Some of these routing problems include scheduling. Due to the significant cost of fuel, cost efficient routes often also end up reducing the CO_2 emission.

When a transport company routes its vehicles or other transporting objects to satisfy a set of specified demands the problem often includes time and/or capacity constraints. In those cases there is a problem of assigning the demands to the vehicles to satisfy the time or capacity constraint. Routing problems for which an assignment of demands occurs are called routing and scheduling problems.

There are many different versions of the network design problem (for examples see [12], [15] and [16]). Magnanti and Wong [15] show how a large set of routing problems can be considered as network design problems by presenting a general model. Even though Magnanti and Wong in [15] show that their model can be used to solve the shortest path problem, the minimum spanning tree problem and others, the model has limitations when it comes to non-positive costs. The model in [15] also has limitations when dealing with pickup and delivery problems. We here choose to simply formulate network design as:

Network design a sub set of connections satisfying some requirements.

The requirements to be satisfied in the network design problem can be to select a set of links connecting a given set of points to each other or find a set of connections which generates a path through a given set of points.

In real-life, network design problems often include:

Routing selection of a path by which an object will travel.

Scheduling assignment of tasks or commodities to objects often including a time aspect.

Note that in practise the term network design is often used for a complete set of routes on a fixed schedule for an extended period of time satisfying some combined requirements.

Scheduling can be used for time dependent routing where a schedule for when the vertices are visited is created and it is often used for connections which have a limited capacity.

Many things can be routed such as the commodity of a demand, vehicles, vessels, cables, railway tracks and water pipes. Routing can also be to select a path of travel for a demand on a predesigned network. However, when there are several demands to be routed the routing of the demand also involves scheduling. The scheduling becomes nontrivial when capacity or time is constrained.

In the remainder of Part I the routing problems are separated into two categories:

Transportation object routing: the routing of transportation objects such as vehicles, vessels, cables and personnel.

Demand routing: the routing of the commodities of the demands on a given network. The given network can be transportation objects carrying the demand.

When optimizing routing and scheduling, knowledge of the demand is important especially in problems where increasing the capacity of the network is costly or time requirements are tight. Finding an attractive route for a demand on a predesigned network will aim at low cost and a good service level. A low cost may result in increased profit or more competitive prices. A good level of service often involves the time of arrival.

For some real-life problems the demand is known in advance. This can be companies where the orders are placed before the routing and scheduling takes place, or situations where the demand is very predictable such as garbage collection.

However, for other types of problems there is an uncertainty of demand. This may be due to frequent disruptions in the plan or that the routing and scheduling is made for a time period where the demand is not determined yet. In the latter case a forecast of the demand is often used when solving the routing and scheduling problem.

For the routing problem it is important to consider what type of objects are transported as the type of object transported often will have influence on the objective and sometimes generate additional constraints.

In the literature the demand is generally separated into two general types which often have different objectives:

Passenger transport For passenger transport the cost is obviously important but the travel time and availability is also part of the objective to ensure a constant or increasing demand.

Freight transport For freight the cost is very important, however for perishable goods or on demand freight there may be some requirement to the transportation time.

For some routing problems two different problems are considered depending on whether it is people or freight cargo which is transported (see [18] and [19]).

Naturally passenger and freight transport problems can both be separated into many more demand subgroups which have specific criteria attached to them.

For routing and scheduling transport objects, where all the demand is required to be delivered, a design is made which minimizes the cost. Since the fuel prices often are a significant cost, reducing the cost also reduces the CO_2 emission. Therefore solving the routing and scheduling problems is often both attractive to the transportation companies and the environment. However, this correspondence between reducing cost and CO_2 emission is unfortunately often not present when changing the form of transport to a less polluting transport form as this often requires new investments. Therefore, environmental organizations should have more focus on supporting the area of finding efficient routes.

1.1 Contribution

The purpose of this thesis is to study some complex routing problems occurring in different situations. Different solution methods are applied to the problems and the aim is to apply the method suiting the problem and circumstances in which the problem needs to be solved. Naturally it has not been possible to cover all solution methods and routing problems. Therefore the problems presented in this thesis have been a somewhat scattered picking of interesting routing and scheduling problems. The problems have been selected mostly based on their novel properties rather than an overall type or solution method. Applying many different methods has also resulted in a less detailed exploration of the methods and many may be improved by further research and exploration. In the remainder of this thesis several different problems are modeled and solution methods are developed for the considered routing problems, each based on several different concepts. The routing problems considered are listed in the following section. Table 1.1 contains a list of solution methods presented in the research papers of Part II and a short description of the problems, the methods are applied to. These problems are among others multi-objective non-additive shortest path problems, different generalizations of the vehicle routing problem including pickup and delivery problems and dial-a-ride problems. The special properties considered in these problem include synchronization, multiple objectives, split-delivery and many others.

Solution method	Problem type	Special conditions	Chapter
Dynamic programming (Exact)	Multi-objective non-additive shortest path problem	Multiple objectives, non-additive criteria Pareto optimal solutions	Chapter 4
Simulated Annealing (Heuristic)	Dial-a-ride problem	Synchronization of routes at points, solution needed within minutes	Chapter 6
Branch-and-cut (Exact)	Pickup-and-delivery problem	Liner shipping network design, transhipment allowed, multiple depots	Chapter 5
Branch-cut-and-price (Exact)	Vehicle routing problem	Capacity and time window constraints, edge sets with one time accessing cost	Chapter 7

Table 1.1: A table over solution methods applied indicating the problems they are applied to.

1.2 Outline

The remainder of this thesis is organized as follows: In Chapter 2 some general models for a few different routing problems are presented. These routing problems can be considered as a basis for the problems presented in the papers in Part II. In Chapter 3 the content of each paper, presented in Part II of this thesis, is summarized. In Part II each chapter consists of a paper. The papers in Part II constitute the major part of the work in this thesis. Chapter 4 covers special cases of the shortest path problem. Chapter 5 describes a model and solution method for finding routes and schedules for liner shipping and Chapter 6 describes a problem of transporting passengers with reduced mobility in an airport. Chapter 7 discusses a special case of the vehicle routing problem with time windows where the edges might belong to a group with a one time cost to be paid to gain access to the edges of the of group. Chapter 8 describes properties of the Euclidean Steiner network. Chapter 9 describes a new way to decompose the vehicle routing problem with time windows. Finally in Chapter 10 we conclude on the findings presented in the papers of Part II.

Chapter 2

Routing and scheduling problems

In this chapter we will present some standard routing problems. The routing problems presented here are related to the various routing problems formulated and solved in Part II of this thesis. The routing problems in Part II are based on real-life applications and therefore include different non-standard real-life constraints or objectives. However the problems presented in Part II can be viewed as generalizations of the different standard routing problems presented here.

The routing problems are defined on a network. Usually the network is represented as a graph $G(V, A)$ where V is the set of vertices in the graph G and A is the set of arcs in G connecting the vertices in V. When the arcs are undirected they are called edges and their set is denoted by E. Let $y_{ij} \in \{0, 1\}$ be an indicator variable which is 1 if and only if the arc or edge connecting i to j is selected. Let $y' = \{(ij) \in A | y_{ij} = 1\}$ and let $S \subseteq V$ be a set of vertices.

Some routing problems can be solved in polynomial time, however with additional constraints and complex objectives even the simple routing problems often become NP-hard. The representation of the graph can also have a great influence on solving the routing problems in practise. In Section 2.1 specific routing problems without scheduling are discussed. While a selection of standard routing and scheduling problems are presented in Section 2.2.

2.1 Routing problems

In this section, some basic routing problems such as path and spanning network problems are presented.

2.1.1 Paths

A routing problem can be to select a path from one vertex to another on a graph $G(V, A)$. A path from vertex o to vertex h can be defined as a sequence of vertices in G starting with o and ending with h with arcs in A from each vertex in the sequence to the next vertex.

Let $S_o \subseteq V \setminus \{h\}$ be a set of vertices containing o and not t then $V \setminus S_o$ contains h and therefore a selection of arcs y' contains a path from o to h if:

$$\sum_{i \in S_o} \sum_{j \in V \setminus S_o} y_{ij} \geq 1, \qquad \forall S_o \subset V \setminus \{t\} : o \in S_o \tag{2.1}$$

To avoid subtours and arcs connected to the path but not on the path we ensure that

$$\sum_{j \in V} y_{ij} \leq 1, \qquad \forall i \in V \tag{2.2}$$

and

$$\sum_{j \in V} y_{ji} \leq 1, \qquad \forall i \in V \tag{2.3}$$

To ensure that the selection y' does only contains arcs connected to the path from o to h a connectivity constraint is needed:

$$\sum_{i \in V} y_{il} \leq \sum_{i \in S} \sum_{j \in V \setminus S} y_{ij}, \qquad \forall \{o,h\} \subset S \subset V, l \in V \setminus S \tag{2.4}$$

A minimizing path model can then be formulated as:

$$\begin{aligned}
\min \quad & f(y) && && (2.5)\\
s.t. \quad & \sum_{i \in S_o} \sum_{j \in V \setminus S_o} y_{ij} \geq 1, && \forall S_o \subset V \setminus \{h\} : o \in S_o && (2.6)\\
& \sum_{i \in V} y_{il} \leq \sum_{i \in S} \sum_{j \in V \setminus S} y_{ij} && \forall \{o,h\} \subset S \subset V, l \in V \setminus S && (2.7)\\
& \sum_{j \in V} y_{ij} \leq 1 && \forall i \in V && (2.8)\\
& \sum_{j \in V} y_{ji} \leq 1 && \forall i \in V && (2.9)
\end{aligned}$$

Where f is a minimizing function of y. Note that there is a exponential number of constraints (2.6) and (2.7). In this formulation negative arc weights and negative cycles can be present. When negative cycles can be present the problem is NP-Hard as it can be shown that the well known NP-complete Hamiltonian path problem is a special case of the elementary shortest path problem. When negative cycles are not present the elementary shortest path problem can be solved in polynomial time by a shortest path problem algorithm which does not remove subtours.

There exists many different path problems and the most commonly applied is the shortest path problem and various generalizations of this problem. The shortest path problem is the problem of finding a route from a point A to a point B on a network minimizing the length of the route. This problem can when the arc costs are nonnegative be solved in $O(V \log V + E)$ time using dynamic programming [4]. However, with negative arc costs it is necessary to eliminate subtours created by negative cycles. Negative arc costs and introducing additional resource constraints can make the problem NP-Hard, see [6]. Moreover, the shortest path problem can when containing several objectives and thereby requiring the set of Pareto optimal solutions, in some cases even be intractable.

Shortest path problems with multiple criteria and objectives are investigated in Chapter 4 while the elementary shortest path problem is used as the subproblem part of the branch-cut-and-price solution method for the vehicle routing problem with time windows and edge set costs presented in Chapter 7.

2.1.2 Spanning trees

Spanning trees are defined on undirected graphs. A spanning tree is a selection of edges y' in $G(V,E)$ so that for any two vertices o and h in V there is a path between o and h in y'.

Let $S \subset V$ be a non-empty set of vertices, then y' contains a path between any two pairs of vertices in V if:

$$\sum_{e \in \{i,j | i \in S \wedge j \in V \setminus S\}} y_e \geq 1, \qquad \forall \emptyset \subset S \subset V \tag{2.10}$$

Since a tree does not contain cycles the following constraint is needed:

$$\sum_{e \in E} y_e \leq |V| - 1 \tag{2.11}$$

A minimum spanning tree model can then be formulated as:

$$\min \quad f(y) \tag{2.12}$$

$$s.t. \quad \sum_{e\in\{i,j|i\in S\wedge j\in V\setminus S\}} y_e \geq 1, \qquad \forall \emptyset \subset S \subset V \tag{2.13}$$

$$\sum_{e\in E} y_e \leq |V| - 1 \tag{2.14}$$

Where f is a minimizing function of y. Note that constraint (2.14) may not be needed if each edge has a nonnegative cost. The constraints (2.13) are similar to the constraints (2.6) in the shortest path problem. The difference is that the spanning tree model applies to all vertices $o, h \in V$ whereas for the path model it is a specific o and h. Clearly there is an exponential number of constraints (2.13). However the spanning tree problem is solvable in polynomial time using for example Prim's or Kruskals minimum spanning tree algorithms.

2.1.3 Arborescence

An arborescence is a rooted directed spanning tree. In an arborescence there exists a single path from a given root vertex o to any other vertex in the graph. This can on a graph $G(V, A)$ be formulated as:

$$\sum_{i\in S_o}\sum_{j\in V\setminus S_o} y_{ij} \geq 1, \qquad \forall\{o\} \subset S_o \subset V \tag{2.15}$$

Since it is required that the set y' representing the arborescence contains no cycles the following constraint is needed:

$$\sum_{i\in V}\sum_{j\in V} y_{ij} \leq |V| - 1 \tag{2.16}$$

A minimum arborescence model can then be formulated as:

$$\min \quad f(y) \tag{2.17}$$

$$s.t. \quad \sum_{i\in S_o}\sum_{j\in V\setminus S_o} y_{ij} \geq 1, \qquad \forall\{o\} \subset S_o \subset V \tag{2.18}$$

$$\sum_{i\in V}\sum_{j\in V} y_{ij} \leq |V| - 1 \tag{2.19}$$

where f is a minimizing function of y. Again the formulation is exponential with an exponential number of constraints of type (2.18). However the minimal arborescence can be formulated with a polynomial number of constraints:

$$\min \quad f(y) \tag{2.20}$$

$$s.t. \quad \sum_{i\in V} y_{ij} \geq 1, \qquad \forall j \in V \setminus \{o\} \tag{2.21}$$

$$\sum_{i\in V\setminus\{o\}} y_{oi} \geq 1, \tag{2.22}$$

$$\sum_{i\in V}\sum_{j\in V} y_{ij} \leq |V| - 1 \tag{2.23}$$

The minimal arborescence can as the minimum spanning tree be found in polynomial time see [9].

2.1.4 n-connected network

A further extension of the spanning tree is the n-connected network problem. A n-connected network is a selection of edges y' on a graph $G(V, E)$ in which there are at least n edge disjoint paths between any two vertices in V. The model is the same as for the spanning trees where the single path between each pair of vertices is replaced by n edge disjoint paths. As a result of this the 1 on the right hand side in the constraints (2.13) is replaced by an n:

$$\min \quad f(y) \tag{2.24}$$

$$s.t. \quad \sum_{e \in \{i,j | i \in S \wedge j \in V \setminus S\}} y_e \geq n, \qquad \forall \emptyset \subset S \subset V \tag{2.25}$$

where f is a minimizing function of y. The n-connected network problem is NP-Hard as the decision problem is a special case of the hamiltonian cycle problem where $n = 2$ and all edges has the same cost. This result may seem surprising when the 1-connected network termed the minimum spanning tree can be found in polynomial time. Note that there is an exponential number of constraints of type (2.25). Moreover note that this formulation does not eliminate extra edges introduced by negative edge costs. A special case of a 2-connected network optimization problem is the traveling salesman problem (TSP), where the problem is to select a path that passes through each vertex exactly once and terminates in the vertex it started in. The TSP can be formulated as a 2-connected network problem where exactly 2 edges in y' are connected to each vertex. Since each vertex is visited exactly once there needs to be added an additional constraint for this requirement.

$$\min \quad f(y) \tag{2.26}$$

$$s.t. \quad \sum_{e \in \{i,j | i \in S \wedge j \in V \setminus S\}} y_e \geq 2, \qquad \forall \emptyset \subset S \subset V \tag{2.27}$$

$$\sum_{e \in \{i,j | j \in V\}} y_e \leq 2, \qquad \forall i \in V \tag{2.28}$$

where f is a minimizing function of y. The TSP is NP-Hard. Note that negative edge costs does not have any influence on the model presented for the TSP. The constraints (2.27) ensure that there is at least two edges connecting any two subsets of V. The constraints (2.28) ensure that every vertex is at most visited once. Combined, (2.27) and (2.28) ensure that every vertex is visited exactly once in a connected route.

2.1.5 Steiner networks

Steiner networks are networks where only a subset of the vertices V on a graph $G(V, E)$ are required to be connected. The set of vertices V are separated into two sets one containing required vertices R and another containing the optional vertices N. The division of the vertices V into required vertices R and optional vertices N introduces restrictions on the sets S used in (2.25). Otherwise the formulation is identical to that for the n connected networks.

$$\min \quad f(y) \tag{2.29}$$

$$s.t. \quad \sum_{e \in \{i,j | i \in S \wedge j \in V \setminus S\}} y_e \geq n, \qquad \forall S \subset V, S \cap R \neq \emptyset, (V \setminus S) \cap R \neq \emptyset \tag{2.30}$$

where f is a minimizing function of y. A specific type of Steiner network is the Euclidean Steiner network. In this network, the optional vertices N are the points in the plane. Even though for the minimal Euclidean Steiner network it is known that the Steiner points all have exactly three edges connected and the angle between two edges is 120^o it is NP-Hard to find the minimal

Steiner tree [14]. The most promising current methods use the construction of full Steiner trees (FST) which are Steiner trees where all required points have exactly one edge connected. For any combination of required vertices the FST is unique [8]. An efficient exact algorithm for solving the minimal Steiner tree problem has been developed by Warme et al. [20]. Euclidean Steiner trees can be used when planning wiring for different purposes such as data transfer. In the wiring problems it is sometimes an issue to have reliable networks. Clearly, if a connection between two vertices is broken then there may be many vertices which are disconnected and the network will not function. A way to improve reliability is to have a 2-connected edge disjoint network. In Chapter 8, structural properties for finding the minimal 2-connected Euclidian Steiner network are described.

2.2 Routing and scheduling problems

In the problems described until now the demands have been the connections and an actual commodity to be transported on the designed networks is not considered. Once we impose a resource constraint such as distance from root, capacity, or time windows, the problems often becomes much harder to solve as they involve scheduling. Scheduling is, as defined in Chapter 1.2, the assignment of tasks to objects. In problems where scheduling is required, knowledge of the demand is essential. The demand must be assigned to routes or edges so that the desired service level is obtained. In contrast to spanning network problems the routing and scheduling problems are mainly applied to directed graphs.

In routing problems where the demand is transported in moving objects it is often less costly to change a routing plan than to change the capacity on links by increasing the capacity of a vehicle. For routing and scheduling problems the demand is often known when the routing and scheduling is done. For some problems where the transporting objects must follow a published time schedule for a given time horizon the demand may not be known but is instead forecasted.

For some problems the demand must be satisfied while the cost is minimized and for other problems the objective is a measurement of how well the demand is satisfied. Again others are a combination of the two.

In this section selected routing problems requiring scheduling are described.

2.2.1 The vehicle routing problem

One of the most researched routing problems which include scheduling is the vehicle routing problem (VRP), see [1], [5], [11] for recent research and [13] for a survey on VRP with time windows. The VRP includes a set of vehicles K, a depot d where all vehicles starts and end their tour and a set of customers C to be serviced by the vehicles. The problem is to route the vehicles so that all vertices are visited exactly once. The most frequently considered demand problem is the capacitated VRP (CVRP) where each customer has a demand which must be delivered from the depot to the customer. It is important to note that in the CVRP it is assumed that the demand is for all customers delivered from the depot to the customer. Therefore the accumulated demand of the customers visited by a vehicle can not exceed the capacity of the vehicle. In the CVRP it is assumed that all vehicles have the same capacity. Another common VRP with scheduling is the problem where the customer must be serviced within a specified time window. This problem is called the VRPTW. Often the VRPTW will include capacity constraints as well. Note that both the CVRP and the VRPTW include scheduling. It should be noted that the VRP problem is a generalization of the bin packing problem (BPP) which is NP-hard in the strong sense.

For the VRP we have a graph $G(V, A, K)$. In the model presented here the depot is modeled as two vertices, a start and an end vertex. The problem is then to find a path for each vehicle starting and ending at the depot so that all customers are serviced by some vehicle and the additional requirements such as time windows and capacity are satisfied. Let o represent the start at depot d and h the end at depot d.

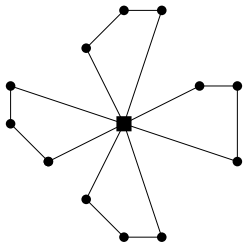


Figure 2.1: A vehicle routing problem where the depot is represented by a square and the filled circles are costumers which each have a demand. Such a demand can be goods to be delivered. Each loop represents a vehicle servicing a set of costumers.

First, it is needed to ensure that the paths visit each customer exactly once.

$$\sum_{k\in K}\sum_{j\in V} y_{ij}^k = 1, \qquad \forall i \in C \tag{2.31}$$

Since each customer must at most be visited once, it is not possible for an arc to be used by several vehicles nor by the same vehicle multiple times. This is however not a problem as the VRP is defined on complete graphs. Note, that constraints (2.31) also eliminates the possibility of subtours at customers. Next, we need to ensure that for each k the arcs selected form an elementary path from o to h, this can be done by using the path constraint on each vehicle:

$$\sum_{i\in S_o}\sum_{j\in V\setminus S_o} y_{ij}^k \geq 1, \qquad \forall k \in K, S_o \subset V \setminus \{h\} : o \in S_o \tag{2.32}$$

To ensure that the vehicles start and end at the depot we introduce constraints:

$$\sum_{j\in V} y_{oj}^k = 1, \qquad \forall k \in K \tag{2.33}$$

and

$$\sum_{j\in V} y_{jh}^k = 1, \qquad \forall k \in K \tag{2.34}$$

To ensure that only arcs connected to the path from o to h are contained in the selection a connectivity constraint is needed:

$$\sum_{i\in V} y_{il}^k \leq \sum_{i\in S}\sum_{j\in V\setminus S} y_{ij}^k, \qquad \forall k \in K, \{o,h\} \subset S \subset V, l \in V \setminus S \tag{2.35}$$

Note that it is assumed that there exists arcs from o to h for every $k \in K$ and that there is no arc entering o as well as no arcs leaving h.

for the CVRP the formulation of the routing problem can be as follows:

$$\min \quad f(y) \tag{2.36}$$

$$s.t. \quad \sum_{k\in K}\sum_{j\in V} y_{ij}^k = 1, \qquad \forall i \in C \tag{2.37}$$

$$\sum_{i\in S_o}\sum_{j\in V\setminus S_o} y_{ij}^k \geq 1, \qquad \forall k \in K, S_o \subset V \setminus \{h\} : o \in S_o \tag{2.38}$$

$$\sum_{j\in V} y_{oj}^k = 1, \qquad \forall k \in K \tag{2.39}$$

$$\sum_{j\in V} y_{jh}^k = 1, \qquad \forall k \in K \tag{2.40}$$

$$\sum_{i\in V} y_{il}^k \leq \sum_{i\in S}\sum_{j\in V\setminus S} y_{ij}^k \qquad \forall k \in K, \{o,h\} \subset S \subset V, l \in V \setminus S \tag{2.41}$$

$$\sum_{j\in V}\sum_{i\in C} y_{ji}^k d_i \leq C_k \qquad \forall k \in K \tag{2.42}$$

where C_k is the capacity of vehicle $k \in K$ and d_i is the demand required to be delivered from the depot to customer i. In the model the constraints (2.42) are introduced ensuring that the accumulated demand of the customers visited by vehicle k does not exceed the capacity C_k of the vehicle.

For the time window version of the VRP (VRPTW) each costumer $i \in C$ must be visited within a customer specific time window $[a_i, b_i]$. Moreover the tour may also need to be completed within a time window. To know the time of the vehicle at the different customers one must keep track of the order in which the customers are visited on the path driven. Therefore it is obvious to let the time window constraint also ensure that the route is connected. A new variable t_i^k is introduced indicating the time at which vehicle k arrives at customer i. Let the travel time of each arc (i, j) be θ_{ij}. Then the following must be satisfied:

$$a_i \leq t_i^k \leq b_i \qquad \forall\, i \in V, k \in K \tag{2.43}$$

$$(t_i^k + \theta_{ij})y_{ij}^k - t_j^k \leq 0 \qquad \forall\ k \in K,\ (i,j) \in A : i \in C \tag{2.44}$$

Note that inequality (2.44) is not linear and the linear version is:

$$(t_i^k + \theta_{ij}) - M_t(1 - y_{ij}^k) - t_j^k \leq 0 \qquad \forall\ k \in K,\ (i,j) \in A : i \in C \tag{2.45}$$

where M_t is a large number greater than or equal to the latest possible finish time. The constraints (2.45) ensure that every cycle formed by y' goes through the depot and thereby forms a connected route. Therefore when introducing constraints (2.45) the constraints (2.41) are redundant and can be removed. Since there is an exponential number of constraints (2.41) it is an advantage to remove these constraints.

As a result the constraints (2.38) can when adding (2.43) and (2.45) be replaced by:

$$\sum_{j\in V} y_{ji}^k - \sum_{j\in V} y_{ij}^k = 0 \qquad \forall k \in K, i \in C \tag{2.46}$$

This will ensure a polynomial number of constraints in the formulation. However the constraints (2.45) in the polynomial formulation are big "M" constraints and weaker than the constraints (2.41).

In Chapter 7 a generalization of the VRPTW is covered and in Chapter 9 an alternative solution method strategy for the VRPTW is considered.

Note that in the version of the VRP considered here the route of the commodity in the demand is indirectly optimized in the assignment of customer vertices to vehicles and the route chosen for each vehicle. In problems where a demand is to be picked up at one customer and delivered to another customer it would require the model to keep track of the demand in the vehicle.

2.2.2 Pickup and delivery problems

In pickup and delivery problems (PDP) a demand is transported between pickup and delivery locations. As can be seen in the survey on pickup and delivery problems by Parragh et al. [18] there are many different types of pickup and delivery problems and most will not be discussed here. The pickup and delivery problems considered in this thesis are problems where each demand is defined by an amount, a pickup location and a delivery location. When there is a homogeneous fleet of vehicles starting and ending their journey in the same location, the problem can be considered to be a generalization of the VRP. In the PDP defined on $G(V, A)$ a variable l_i^k containing the load on vehicle k at vertex i is needed. Moreover it is important to ensure that demand picked up is afterwards delivered. Let there be a set of demands M and let each demand be represented as $m = (o^m, h^m, d^m) \in M$ where o^m is the pickup location of demand m, h^m is the delivery location of m and d^m is units of load of demand m. Let l'_j be the load change at vertex j. Then the PDP can be formulated using the formulation for the VRP (2.36) to (2.40) and (2.45) with additional constraints for load and pickup and delivery requirements. In the following it is assumed that there is a vertex for each demand pickup location and delivery location. Then the additional constraints are:

$$l_i^k + l'_j - M_l(1 - y_{ij}^k) \leq l_j^k \qquad (i,j) \in A,\ k \in K \tag{2.47}$$

$$l_i^k \leq C_k \qquad i \in V,\ k \in K \tag{2.48}$$

$$\sum_{i \in V} y_{io^m}^k - \sum_{i \in V} y_{ih^m}^k = 0 \qquad m \in M, k \in K \tag{2.49}$$

$$t_{d^m}^k - t_{o^m}^k \geq 0 \qquad k \in K,\ m \in M \tag{2.50}$$

Constraints (2.47) ensure that the load on vehicle k when leaving vertex j is at least the load on the vehicle when arriving to vertex j adding the load change l'_j required at vertex j. Note that for a pickup location the load change l'_j is positive and for a delivery location the load change is negative. Constraints (2.48) ensure that the load does not at any point exceed the capacity of the vehicle. Constraints (2.49) ensures that if a pickup vertex o^m of a demand m is visited by a vehicle k then the delivery vertex h^m of the demand is also visited by the vehicle k. Constraints (2.50) ensure that the pickup vertex of a demand m is always visited before the delivery vertex of m.

In some problems the demand may not need to be delivered by one single route. In those cases it might be possible to utilize the capacity of the vehicles better allowing better solutions by splitting up a single demand and transporting the partial demands on different routes. Allowing the splitting up of a single demand is generally called *split delivery* or *split load* see [17]. Moreover in some problems it might be allowed to move some demand over from one vehicle to another so that the vehicle picking up the demand is not the same as the one delivering the demand. This is also called transhipping or cross docking see [2] and [21].

In problems where transhipment and split delivery are allowed the problem of the routing of the demands becomes a multi commodity flow problem which can be formulated as a linear programming problem. In this case a variable x_{ij}^{mk} indicating the flow on each arc is used. We let the parameter l'_{mi} be the change in the flow of demand m at vertex i so that for o^m the l'_{mo^m} is the amount of m to be shipped. Then $l'_{mh^m} = -l'_{mo^m}$ and zero at any other vertex. The constraints keeping track of the demand can look as follows:

$$\sum_{v \in \mathbf{V}} \sum_{j:(i,j) \in \mathbf{A}} x_{ij}^{mk} - \sum_{v \in \mathbf{V}} \sum_{j:(j,i) \in A} x_{ji}^{mk} = l'_{mi} \qquad \forall i \in N, \forall m \in M \tag{2.51}$$

$$\sum_{m \in \mathbf{M}} x_{ij}^{mk} \leq y_{ij}^k C_k \qquad \forall (i,j) \in \mathbf{A}, \forall k \in \mathbf{K} \tag{2.52}$$

$$x_{ij}^{mk} \in \mathbb{R}_0^+ \tag{2.53}$$

where the capacity constraints (2.52) replace the constraints (2.48).

In this case the constraints (2.41) from the VRP are needed to ensure connectivity. In the real-life PDP described in Chapter 5 both transhipment and split delivery are allowed.

2.2.3 Dial-a-ride problems

The dial-a-ride problem (DRP) is a PDP where the objects transported are people. The fact that it is people that are transported generates a set of special constraints. These special constraints are connected to the time aspect. A person generally dislikes spending an excessive amount of time in the vehicle whereas the primary concern of a package is to be delivered on time. Therefore a good solution for freight PDP is often not a good solution for transporting humans. The DRP includes constraints such as a limit on the travel time between pickup and delivery of a person. Moreover the objective is often changed so that a service level is included in the objective by for example minimizing the excess travel time used. As for the PDP the commonly considered case for the DRP is the one where transhipment and split delivery are not allowed since a single demand often corresponds to one person whom can not be split. The constraint added to the PDP is

$$t^k_{h^m} - t^k_{o^m} - \hat{t}^k_m \leq H \qquad k \in K,\, m \in M \tag{2.54}$$

where $\hat{t}^k_m$ is the minimum amount of time required to go from o^m to h^m and H is a limit on the excess time allowed. The constraints (2.54) ensures that a hard time limit is put on the excess time used on the journey form o^m to h^m.

In Chapter 6 a real-life DRP which contains transhipment at predetermined points is presented. An excellent survey on the DRP problem can be found in Cordeau and Larporte [3].

Chapter 3

Summaries of papers included

A summery of each paper is be presented in the order they appear in Part II.

Chapter 4: Multi-Objective and Multi-Constrained Non-Additive Shortest Path Problems.
In this paper, non-additive shortest path paroblems with multiple objectives are studied. A general framework for dominance tests for problems involving a number of non-additive criteria is presented. These dominance tests can help eliminate paths in a dynamic programming framework when using multiple objectives. Results on real-life multi-objective problems containing non-additive criteria are reported. It is shown that in many cases the framework can be used to efficiently reduce the number of generated paths. This paper was publish in *Computer & Operations Research* volume 38, issue 3, March, 2011. A preliminary version of this paper has been published as a technical report at DTU Management.

Chapter 5: A Branch and Cut algorithm for the container shipping network design problem.
A model and solution method for the network design problem in liner shipping is presented. The network design and fleet assignment problems are combined into a mixed integer linear programming model minimizing the overall cost. To better reflect the real-life situation we take into account the cost of transhipment, a heterogeneous fleet, route dependent capacities, and butterfly routes. To the best of our knowledge it is the first time an *exact* solution method considers transhipment cost. The problem is solved with branch-and-cut using clover and transhipment inequalities. Computational results are reported for instances with up to 15 ports. This paper has been conditionally accepted to the *Flexible Services and Manufacturing Journal* special issue on Maritime Container Logistics and Onshore Transportation Systems. A preliminary version has been published as a technical report at DTU Management and has been presented at the Global Maritime & Intermodal Logistics Conference 2007 in Singapore and later version at Optimization days 2010 in Montreal.

Chapter 6: Dial-a-ride with synchronization for handicap assistance at airports
The problem of transporting passengers with reduced mobility in major transit airports is described, modeled and solved. In major airports, a significant number of people and busses are assigned to provide transportation for passengers with reduced mobility. It is often necessary for a passenger with reduced mobility to use several different modes of transport during their journey through the airport. Synchronization occurs at the locations where transport modes are changed as to not leave passengers unattended. A simulated annealing based heuristic for solving the problem is presented. The algorithm makes use of an abstract representation of a candidate solution which in each step is transformed to an actual schedule by use of a greedy heuristic. Local search is performed on the abstract representation using advanced neighborhoods which modify large parts of the candidate solution. Computational results are reported showing that the algorithm is able to find good solutions within a couple

of minutes, making the algorithm applicable for dynamic scheduling. Moreover high-quality solutions can be obtained by running the algorithm for 15 minutes. This paper is submitted to *European Journal of Operations Research*. A preliminary version has been published in a technical report at DTU Management and presented at the 4^{th} Nordic Optimization Symposium.

Chapter 7: The vehicle routing problem with edge set costs.
An important generalization of the vehicle routing problem with time windows in which a fixed cost must be paid for accessing a set of edges is considered. This fixed cost can reflect payment for toll roads, investment in new facilities, the need for certifications and other costly investments. The certifications and contributions impose a cost for the company while they also give unlimited usage of a set of roads to all vehicles belonging to the company. Alternative versions for defining the edge sets are discussed and formulated. A MIP-formulation of the problem is presented, and a solution method based on branch-cut-and-price is applied to the problem. The computational results presented in the paper show that instances with up to 50 customers can be solved in reasonable time, and that the branch-cut-and-price algorithm generally outperforms CPLEX. Another observation is that instances get more difficult when the penalized edge sets form a spanning tree, compared to when they are randomly scattered. This paper has been submitted to *Networks* and preliminary results have been presented at the 24^{th} European Conference on Operational Research 2010.

Chapter 8: Bounding component sizes of two-connected Steiner networks.
In this paper the problem of constructing a shortest Euclidean 2-connected Steiner network in the plane for a set of n required terminals. This problem has natural applications in the design of survivable communication networks. Combining several previous results it is in this paper shown that no full Steiner tree in a two-connected Steiner network spans more than $\lfloor n/3 \rfloor + 1$ terminals. This paper was published in *Information Processing Letters* volume 104, issue 5, 30 November 2007. A preliminary version of this paper has been published in the proceedings of the 13^{th} Computing: The Australasian Theory Symposium (CATS2007), Ballarat, Australia.

Chapter 9: Root Balanced Minimum Spanning Graph: Algorithm and Complexity.
In a cycle-free graph where $V = \{0, 1, \ldots, n+1\}$ is the vertex set, and $E = \{(i,j)|i,j \in V\}$ is the set of edges. The Minimum Root Balanced Spanning Graph Problem is to find a minimum weight subgraph which contains a spanning tree on the vertices 0 to n vertices where the number of edges between the vertex sets $\{1, \ldots, n\}$ and $\{n+1\}$ is equal to the number of edges between $\{0\}$ and $\{1, \ldots, n\}$. This problem is interesting as it appears as a subproblem in some decompositions of the vehicle routing problem. In this paper an algorithm to solve this problem in $O(|E|\alpha(|E|,|V|)+|V|\log|V|)$ time where α is the classical functional inverse of the Ackermann's function is presented. It is proven that the shortest spanning arborescence with time windows is NP-complete. This paper is a working paper and the research on the topic has not been completed before the deadline of this thesis.

Bibliography

[1] R. Baldacci, E. Bartolini, A. Mingozzi, and R. Roberti. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7:229–268, 2010.

[2] P. Chen, Y. Guo, A. Lim, and B. Rodrigues. Multiple crossdocks with inventory and time windows. *Computers & Operations Research*, 33:43–63, 2006.

[3] J. F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573–586, 2006.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press/McGraw-Hill Book Company, 1997.

[5] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle reouting problem with time windows. *Transportation Science*, 42:387–404, 2008.

[6] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42:977–978, 1994.

[7] P. Eveborn, M. Rönqvist, H. Einarsdòttir, M. Eklund, K. Lidén, and M. Almroth. Operations research improves quality and efficiency in home care. *Interfaces*, 39:18–34, 2009.

[8] E. N. Gilbert and H. . Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16:1–29, 1968.

[9] T. S. H. N. Gabow, Z. Galil and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6:109–122, 1986.

[10] W. Jang, H. H. Lim, T. J. Crowe, G. Raskin, and T. E. Perkins. Othe missouri lottery optimizes its scheduling and routing to improve efficiency and balance. *Interfaces*, 36:302–313, 2006.

[11] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56:497–511, 2008.

[12] D. S. Johnson, J. K. Lenstra, and A. H. G. R. Kan. The complexity of the network design problem. *Networks*, 8:279–285, 1978.

[13] B. Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35:2307–2330, 2008.

[14] D. S. J. M. R. Garey, R. L. Graham. The complexity of computing steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32:835–859, 1977.

[15] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18:1–55, 1984.

[16] M. Minoux. Optimum network design models and algorithms in transportation and communication. *International Journal of Logistics Research and Applications*, 6:5–15, 2010.

[17] M. Nowak, . Ergun, and C. C. W. III. Pickup and delivery with split loads. *Transportation Science*, 42:32–43, 2008.

[18] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58:81–117, 2008.

[19] M. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

[20] D. Warme, P. Winter, and M. Zachariasen. Geosteiner: Software for computing steiner trees. *http://www.diku.dk/hjemmesider/ansatte/martinz/geosteiner/*, 2003.

[21] M. Wen, J. Larsen, J. F. Cordeau, and G. Laporte. Vehicle routing with cross-docking. *Journal of the Operational Research Society*, 60:1708–1718, 2009.

Part II

Research Papers

Chapter 4

Multi-Objective and Multi-Constrained Non-Additive Shortest Path Problems

Line Blander Reinhardt* David Pisinger*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
lbre@man.dtu.dk, pisinger@man.dtu.dk

Shortest path problems appear as subproblems in numerous optimization problems. In most papers concerning multiple objective shortest path problems, additivity of the objective is a de-facto assumption, but in many real-life situations objectives and criteria, can be non-additive. The purpose of this paper is to give a general framework for dominance tests for problems involving a number of non-additive criteria. These dominance tests can help eliminate paths in a dynamic programming framework when using multiple objectives. Results on real-life multi-objective problems containing non-additive criteria are reported. We show that in many cases the framework can be used to efficiently reduce the number of generated paths.

Keywords: Multi objective programming, Shortest path problem, Non-additive objective, Dynamic programming

4.1 Introduction

The *shortest path problem* can be formulated on a directed graph $G = (V, E)$ where V is a finite set of vertices and E is a finite set of edges. The problem is to find a shortest path between a *source* $s \in V$ and a *destination* $t \in V$. In the *multi-objective shortest path problem* there are r criteria. An edge $e_{ij} \in E$ from vertex $i \in V$ to vertex $j \in V$ has associated values $c_{ij}^k, \quad k \in \{1, ..., r\}$ for each criterion $k = 1, \ldots, r$. In order to have a well-defined problem it is assumed that there are no negative cycles for the criteria being minimized, or positive cycles for the criteria being maximized.

The general *additive* multi-objective shortest path problem with positive costs c_{ij} on the edges and where there is an additive objective function for each criterion can be described by the following

integer model (Martins [18]):

$$\text{minimize}: \quad z = \left(\sum_{(i,j)\in V} c_{ij}^1 e_{ij}, \quad \dots \quad , \sum_{(i,j)\in V} c_{ij}^r e_{ij} \right) \tag{4.1}$$

$$s.t. \quad \sum_{j\in V} e_{sj} - \sum_{j\in V} e_{js} = 1 \tag{4.2}$$

$$\sum_{j\in V} e_{tj} - \sum_{j\in V} e_{jt} = -1 \tag{4.3}$$

$$\sum_{j\in V} e_{ij} - \sum_{j\in V} e_{ji} = 0 \quad \forall i \in V \setminus \{s,t\} \tag{4.4}$$

$$e_{ij} \in \{0,1\} \quad \forall (i,j) \in E \tag{4.5}$$

Constraint (4.2) states that there must be exactly one edge leaving s that is not on a cycle. Constraint (5.3.2) states that there must be exactly one edge entering t that is not on a cycle. Constraint (4.4) is the ordinary flow conservation constraint. The solution z is an r vector which contains the values of the r objective functions for the path.

In real-life problems the objective functions may be *non-additive* and they may be functions of several criteria. In general the objective function has the following form:

$$\text{minimize}: z = \big(C^1(\mathcal{P}), \dots, C^r(\mathcal{P})\big) \tag{4.6}$$

where $\mathcal{P}$ is a path $\{s, \dots, t\}$ and $C^1, \dots, C^r$ are cost functions which for a given path $\mathcal{P}$ return a real number. We call this the *value vector*. A solution to the above problem is a set of all Pareto optimal paths. A path $\mathcal{P}$ is *Pareto optimal* if there is no other path $\mathcal{P}'$ between the same two vertices which is better or equal on all entries of the *value vector* and where at least one entry is better. The solution set where there is exactly one path for each Pareto optimal *value vector* is called the *minimal complete* set of Pareto optimal solutions.

Shortest path problems are among the most well-studied problems [1], however, results concerning multi-criteria problems are rare. This may be due to the fact that the monotonicity assumption of dynamic programming seldom holds for these problems. In [4] Carraway et al. describe monotonicity as the property of objectives preserving preferences for partial solutions in the dynamic programming recursion. This is not to be confused with non-decreasing or non-increasing functions. In shortest path problems the monotonicity criterion means that if $\mathcal{P}$ is a shortest path then the subpaths of $\mathcal{P}$ must also be shortest paths. For real life multi-criteria problems monotonicity only holds for special cases.

One of the reasons for the recent interest in multi-objective optimization is that optimization is being applied in public services and business applications. Therefore research in multi-objective shortest paths and attempts to circumvent the monotonicity assumption is of immediate interest.

Hansen [10] presented solution methods to monotone bicriteria and biobjective path problems using a label setting algorithm. Martins [18] generalized the label setting algorithm to an arbitrary number of objectives, however, monotonicity was still assumed. Brumbaugh-Smith and Shier [3] presented a label correcting algorithm to the multi-objective problem under the monotonicity assumption. As Ehrgott and Gandibleux mention [8], these problems were not extensively researched before the nineties. Several recent papers discuss approximation algorithms such as the FPTAS outlined by Tsaggouris and Zaroliagis [25]. For the exact solution methods Tsaggouris and Zaroliagis have studied nonadditive paths with a single objective [24]. This objective is the sum of several criteria weighted by linear or non-linear coefficients. An early paper by Lengauer and Theune [17] mentions the problem of non-monotone cost structures with two criteria and shows that by changing the domination criteria the problems can be solved by using a standard shortest path algorithm such as Dijkstra's. Another paper considering a non-additive weighted objective is Carraway et al. [4] where the distance is minimized, and the probability of successfully reaching the destination is maximized. We will later show that both the probability and the

distance criteria are monotone even though the sum of the two is not. It is shown in Carraway et al. [4] that it can be difficult to determine whether a multicriteria objective function is monotone. To circumvent the problem of not satisfying the monotonicity assumption Carraway et al. [4] introduced the concept of *generalized dynamic programming*, and presented a framework for solving multi-criteria shortest path problems. The framework, however, leaves it to the concrete application to define a local preference relation, that can be used to remove dominated states in the dynamic programming recursion. In this paper we present a number of such preference relations for the listed criteria functions. Moreover we give a general framework for how to define a local preference relation for non-additive objectives with certain general properties.

The contribution of this paper is to present a number of different criteria functions motivated by real-life applications and to develop an algorithm which finds all Pareto optimal solutions for a multi-objective shortest path problem. We present a general framework for dominance tests with all the presented criteria functions f and report computational experiments on a real-life instance with multiple criteria (addition, maximization, multiplication). Finding all Pareto-optimal solutions to a multi-objective shortest path problem has several similarities with the solution of multi-constrained shortest-path problems [6, 9, 13, 14, 15, 21, 26]. The techniques developed in this paper can therefore be applied to several variants of multi-constrained shortest-path problems with non-additive constraints.

The paper differs from previous work as follows: Müller-Hannemann and Schnee [19] solve the nonadditive price problem by generally relaxing the Pareto optimality. The algorithm by Carraway et al. [4] only determines the *minimal complete* set of the Pareto optimal paths, and therefore does not find all Pareto optimal paths. Carraway uses a function u which maps all criteria to a real number and thereby only has one objective function. His framework does not specify how the dominance function should be implemented, but leaves it open to the concrete application. Irnich and Villeneuve [15] presented an algorithm for finding all Pareto-optimal solutions to a resource constrained shortest path problem with k-cycle eliminations. The considered (constrained and unconstrained) resources are all additive.

In the following Section 4.2, we present a number of multi-objective shortest path problems encountered in practice. Next, in Section 4.3, we formally define the set of Pareto optimal solutions and the corresponding dominance criterion. In Section 4.4, we present the fundamental dynamic programming algorithm used to solve the multi-objective shortest path problem for various cost functions. Section 4.5 provides a number of dominance rules for various cost functions that can be used to prune labels in a dynamic programming algorithm. Various monotone as well as non-monotone cost functions are considered. In Section 4.6, we return to the problems considered in Section 4.2 and discuss how the framework developed can be used to determine all Pareto-optimal solutions. Finally, Section 4.7 reports on computational experiments on real-life data from a shipping company. The paper concludes in Section 4.8.

4.2 The Multi-Objective Shortest Path Problems

Multi-criteria shortest path problems are well-studied for additive objective functions. However, in several real-life settings one cannot assume that the objective function is additive, neither can one assume that it is monotonously increasing. Let $w : E \to \mathbb{R}$ be an additive weight function on the edge weight and let f be a function from real numbers to real numbers, $f : \mathbb{R} \to \mathbb{R}$. The objective functions we will consider often contain the function $f \circ w$. The following list describes a number of objective functions which might be encountered in real life multi-criteria and multi-objective problems. Note that all of the objectives listed are non-additive:

A *Probability of reaching destination.* Carraway et al. [4] consider the objective of maximizing the probability of successfully reaching the destination. Each edge has an associated cost (length) and a probability for successful traversal. Assuming that probabilities are independent across edges, the probability of successfully reaching the destination is the product of the probabilities of the edges traversed. In Section 4.5, we will show how the product of probabilities can be converted to a function of the form $f \circ w$.

B *Combined distance and probability function.* In [4] Carraway et al. consider an objective that is a combination of the distance d and the probability p. The objective is described as $-d + \lambda p, \quad \lambda \in \mathbb{R}_+$, where the aim is to maximize the objective and thereby minimize the length of the journey and maximize the probability. However, if the two criteria contradict each other the value of λ will affect how the two criteria are weighted. Referring to the objective of real life problem A, it is easy to see that this objective function will be of the form $-w_1 + f \circ w_2$ where $w_1, w_2 : E \to \mathbb{R}$ are additive.

C *Maximum of commissions.* Blander Reinhardt [2] describes a real life multi-objective cargo transportation problem in which each vertex corresponds to a hub port. Each time a vertex is visited (i.e. the cargo is reloaded) an agent is paid a commission. An agent may be responsible for several hubs, but will only be paid one commission. The commission paid will correspond to the largest commission the agent is entitled to on the path. The cost of a path is then the price of the edges plus the commission paid to agents. However, only paying the agents the *largest* commission on the path complicates the objective. The objective function for the price objective is then $w + \sum_{a \in Agents} \max_a \{c_e^a | e \in E_P\}$, where E_P is the set of edges visited on the path $\mathcal{P}$ and c_e^a is the commission paid to agent a on edge e.

D *Number of zones visited.* Public transport in e.g. Copenhagen operates on a zone system [11]. Each zone covers a number of vertices (stations) and edges, and if a ticket is issued to a given zone, then it gives unlimited access to all vertices and edges in the zone. In other words our cost function implies that a cost is paid only the first time a zone is visited. A holder of a *monthly card* may buy access to any zone needed, hence the objective is to minimize the number of different zones on the path. Here the problem is that a price is only paid the first time an edge in the given zone is visited. The objective function can be represented as $\sum_{z_h \in Zones} \max\{0, w_e | e \in E_{Z_h} \wedge e \in P\}$, where E_{Z_h} is the edges in the zone Z_h and $\mathcal{P}$ is the path travelled. The value w_e is 1 for all e.

E *Maximum zone distance from origin.* The cost of a *single-trip ticket* in public transport may depend on the maximum zone distance from the origin s, as in Copenhagen [11]. Here the vertices and the edges again belong to a zone. A one-zone ticket gives access to the zone containing s. A two-zone ticket gives access to all zones adjacent to the first zone. A three-zone ticket furthermore gives access to all zones adjacent to the previous zones. This means that if the non-starting zones visited on a trip all are neighboring to the starting zone s then a two zone ticket is needed even though more than two different zones maybe visited. The objective is to minimize the maximum zone distance between the zone of the edges on the path and the origin s to the destination t. In this case the objective function is as in D. However the zones are defined differently (see Section 4.6).

F *Zone distance and time.* In several public transportation ticket fare systems, one buys access to some zones in a given time period, hence it may be relevant to take both distance and time into consideration. A 1-zone ticket can be traversed within a time limit t_1, a 2-zone ticket can be traversed within a time limit $t_2 \geq t_1$, etc. The objective is to minimize travel cost. In this case the cost function takes a time and a number of zones and returns a cost. Here the objective function *obj* takes the maximum of a time function t and a zone function z, $obj = \max\{t, z\}$, where the zone function is as described in E and the time function t is of the form $f \circ w$.

G *Modulo k penalties.* Jepsen et al. [16] consider a shortest path problem where an additional penalty cost is paid for each of the k times nodes from a given set S have been visited. The objective is the cost of the edges plus a penalty depending on the number of times nodes from the set S have been visited. Here, the complicating factor is the penalty. This shortest path problem stems from the addition of Subset-Row inequalities [16] in a branch-and-price algorithm for the Vehicle Routing Problem. The objective function is as in B, $w_1 + f \circ w_2$, where f is the penalty function.

In the following sections we describe a general framework for solving multi-criteria shortest path problems. Then in Section 4.5 we develop general schemes for non-additive functions of the form $f \circ w$. In Section 4.6, we return to the above problems and discuss how they can be solved using the schemes developed in Section 4.5.

4.3 Pareto Optimal Paths and Value Vectors

A path from s to t is denoted $\mathcal{P}_{st} = \{s, \ldots, t\}$. A sub-path $\mathcal{P}_{ij}$ of $\mathcal{P}_{st} = \{s, \ldots, i, \ldots, j \ldots, t\}$ is the path $\{i, \ldots, j\}$.

The optimal solution to the multi-objective shortest path problem (4.1) is a set of all Pareto optimal paths. A path is *Pareto optimal* if the value vector of that path is not *dominated* by the value vector of another path between the same two vertices. Let $x = (x_1, ..., x_r)$ and $y = (y_1, ..., y_r)$ be two real valued r vectors then for a minimization problem x *dominates* y if $x_k \leq y_k$ for all $k \in \{1, \ldots, r\}$ and $x_k < y_k$ for at least one $k \in \{1, \ldots, r\}$.

The *set of Pareto optimal paths* from a source s to a destination t is the set of paths from s to t with non-*dominated* value vectors. It should be noted that there can be several Pareto optimal paths with the same value vector.

If the problem has more than one objective then there can be an exponential number of Pareto optimal paths where each has a unique value vector. In Hansen [10] a graph is presented where there is an exponential number of paths from x_1 to x_n which are all Pareto optimal and have unique value vectors. However, as observed by Müller-Hannemann and Weihe [20], the number of Pareto optimal paths in real-life problems is usually quite small. This observation is also confirmed in the real-life problems we have studied.

In some problems only the set of *minimal complete* Pareto optimal paths are sought. The *minimal complete* set of Pareto optimal paths was defined in [10] for bicriteria problems. The *minimal complete* Pareto optimal paths are as defined in Section 4.1 the set containing exactly one path per Pareto optimal value vector. Note that the *minimal complete* set of Pareto optimal paths also can contain exponentially many paths, see [10]. When a shortest path problem contains only two objective functions the *minimal complete* set of Pareto optimal paths can be found using an integer programming method called the Ranking method. At each iteration of the ranking method a Pareto optimal value vector and a path satisfying the value vector is found (see [8] for further details). However, we have no knowledge of a constructive way of finding the *minimal complete* set of Pareto optimal paths for problems with more than two objective functions.

It should be mentioned that in quite a few real life problems it would be desirable to find not only the *minimal complete* set of Pareto optimal paths but *all* the Pareto optimal paths. This is in particular true in cases where there is a decision maker who selects the most desirable solution. It is reasonable to assume that there are factors unknown to the program and depending entirely on the specific decision maker which have influence on the choice, and therefore two different paths with the same objective values might be viewed differently by the decision maker.

4.4 Dynamic Programming

Dynamic programming relies on the *principle of optimality*. For multi-objective problems where one or more objectives does not satisfy the monotonicity property, one must use the *weak principle of optimality* as defined by Carraway et al. [4]

Principle of optimality An optimal path must be composed of optimal subpaths.

Weak principle of optimality An optimal path must be composed of subpaths that can be part of an optimal path.

Irnich and Villeneuve [15] defined a similar weak principle of optimality for multi-constrained shortest path problems based on the concept of *extensions* $\mathcal{E}(\mathcal{P})$ of a given subpath $\mathcal{P}$.

There are two general dynamic programming algorithms for additive multi-objective shortest path problems which are based on Dijkstra's algorithm for the single objective shortest path problem [5]. These algorithms are the *Label-Setting* algorithm [18] and the *Label-Correcting* algorithm [22]. The Label-Setting algorithm does not allow negative edge costs. The Label-Correcting algorithm does allow negative edge costs but no negative cycles. We will apply the Label-Correcting algorithm in the sequel.

In the pseudocode for this algorithm let $C^1(\mathcal{P}) \dots C^r(\mathcal{P})$ be the cost function of a path $\mathcal{P}$ and let `Merge` be a function which, given two sets of labels, returns only the undominated labels of the union of the two sets. The set Q consists of the vertices with undominated labels that have not yet been used to generate other labels. Each label is given as the tuple $(C^1(\mathcal{P}), \dots, C^r(\mathcal{P}), \text{pred}(\mathcal{P}))$, where $\text{pred}(\mathcal{P})$ is a pointer to the label it was generated from. The label correcting algorithm is outlined in the following pseudocode, inspired by [3] and [22]. In each node v we maintain a list L_v of labels. Clearly, by following the $\text{pred}(\mathcal{P})$ pointers backward from a vertex v to s one gets the subpath the label represents. Thus each label in the lists L_v represents a subpath $\mathcal{P} = \{s, \dots, v\}$ which is not dominated by other subpaths from s to v. The set Q can with advantage be implemented as an ordered list ordered lexicographically.

LABEL-CORRECTING(G, s, t)
1: $L_v \leftarrow \emptyset$ **for all** $v \in V \setminus \{s\}$
2: $L_s \leftarrow \{(0, \dots, 0, \{s\}))\}$;
3: $Q \leftarrow \{s\}$;
4: **while** $Q \neq \emptyset$ **do**
5: $u \leftarrow$ **extract vertex from** Q;
6: **for all edges** e_{uv} **do**
7: $L'_v \leftarrow$ `Merge`$(L_v, L_u \circ \{e_{uv}\})$;
8: **if** $L'_v \neq L_v$ **then**
9: $L_v \leftarrow L'_v$;
10: $Q \leftarrow Q \cup \{v\}$
11: **end if**
12: **end for**
13: **end while**
14: **return** L_v **for all** $v \in V$;

The label correcting algorithm repeatedly extracts a vertex u from the set Q, and for each outgoing edge e_{uv} extends the labels $(C^1(\mathcal{P}_{su}), \dots, C^r(\mathcal{P}_{su}), \text{pred}(\mathcal{P}_{su}))$ in L_u to $(C^1(\mathcal{P}_{sv}), \dots, C^r(\mathcal{P}_{sv}), \text{pred}(\mathcal{P}_{sv}))$. These new labels $L_u \cup \{e_{uv}\}$ are then `Merge`'d together with the old labels L_v of v. The merging eliminates *dominated* labels. If the set of labels at v has been changed during the `Merge`, then v is added to Q. This is repeated until the set Q is empty. The $\circ$ operator in line 7 must match the objective functions, and the dominance criterion used implicit in `Merge` must be tailored to return the undominated labels. We will in the next section propose various sufficient criteria for removing dominated labels.

4.5 Non-Additive Objectives in Dynamic Programming

As mentioned before, the essence of the dynamic programming algorithms on shortest path problems is the monotonicity requirement. The monotonicity requirement ensures that subpaths of a Pareto optimal path are Pareto optimal and therefore the subpaths that are not Pareto optimal can be eliminated from the search. Clearly the additive case is monotone, however there exists other monotone objective functions. Theorem 1 covers a set of objective functions which satisfies the monotonicity requirement. It should be noted that Theorem 1 is not exhaustive, as other objectives may exist that satisfy the monotonicity requirement.

Section 4.5.1 defines a dominance criterion for problems where the objective function is based on two or more additive weight functions. Section 4.5.2 defines dominance criteria for problems where

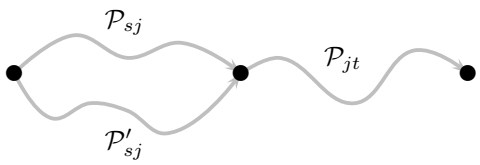


Figure 4.1: Path from s to t is split into $\mathcal{P}_{sj}, \mathcal{P}'_{sj}, \mathcal{P}_{jt}$.

the objective function is defined as the maximum of a set of edge weights visited on the path. The recently published example of optimizing the mean and variance of a random variable associated with and edge described in [12] by Hutson and Shier shows a non-additive objective of the structure described in Section 4.5.1. Weight functions evaluate a single criterion along a path. However an objective can include several criteria. Thus several weight functions including the weight of a criterion on a path can be transformed by one function in the objective. An additive objective requires that the weight functions (also called criteria) included in it are additive, however, an objective on one or more additive weight functions is not necessarily additive as the additive weight function can be part of a non-additive function in the objective.

Theorem 1. *Given a weighted directed graph $G = (V, E)$ with an additive weight function $w : E \to \mathbb{R}^k$ $(w_\ell : E \to \mathbb{R}, \quad \ell \in \{1, .., k\})$ where k is the number of objective functions, let the objective functions be $C_\ell = f_\ell \circ w_\ell$ where $f_\ell : \mathbb{R} \to \mathbb{R}$ is a strictly increasing or strictly decreasing function. Let $\mathcal{P}_{st}$ be a Pareto optimal path from s to t, then any subpath $\mathcal{P}_{ij}$ of $\mathcal{P}_{st}$ is a Pareto optimal path from i to j.*

The theorem shows that if the objective functions are strictly increasing or strictly decreasing, then the normal *principle of optimality* holds, and hence we can use an ordinary dominance rule in the label correcting algorithm.

Proof. First assume that the objective functions are to be minimized. Assume that $\mathcal{P}_{ij}$ is not a Pareto minimal path, then there would be a path $\mathcal{P}'_{ij}$ that dominates the path $\mathcal{P}_{ij}$. Decompose the Pareto minimal path $\mathcal{P}_{st}$ into three subpaths $\mathcal{P}_{si}$, $\mathcal{P}_{ij}$ and $\mathcal{P}_{jt}$. Then, because of the additive structure of the weight function, we have $w(\mathcal{P}_{st}) = w(\mathcal{P}_{si}) + w(\mathcal{P}_{ij}) + w(\mathcal{P}_{jt})$. Let the path $\mathcal{P}'_{st}$ be defined by subpaths $\mathcal{P}_{si}$, $\mathcal{P}'_{ij}$ and $\mathcal{P}_{jt}$. Clearly the weight of $\mathcal{P}'_{st}$ is $w(\mathcal{P}'_{st}) = w(\mathcal{P}_{si}) + w(\mathcal{P}'_{ij}) + w(\mathcal{P}_{jt})$. Since $\mathcal{P}'_{ij}$ dominates $\mathcal{P}_{ij}$ we have $f_\ell(w_\ell(\mathcal{P}'_{ij})) \leq f_\ell(w_\ell(\mathcal{P}_{ij}))$ where the inequality is strict for at least one $\ell \in \{1, ..., k\}$. Then, in the case where f_ℓ is strictly *increasing*, one gets the following inequalities where for at least one ℓ the inequality is strict:

$$f_\ell(w_\ell(\mathcal{P}'_{ij})) \leq f_\ell(w_\ell(\mathcal{P}_{ij})) \quad \Rightarrow \quad w_\ell(\mathcal{P}'_{ij}) \leq w_\ell(\mathcal{P}_{ij}) \quad \Rightarrow$$
$$w_\ell(\mathcal{P}'_{st}) \leq w_\ell(\mathcal{P}_{st}) \quad \Rightarrow \quad f_\ell(w_\ell(\mathcal{P}'_{st})) \leq f_\ell(w_\ell(\mathcal{P}_{st}))$$

In the case where f_ℓ is strictly *decreasing* one gets a similar result by reversing the appropriate inequalities. Therefore the path $\mathcal{P}'_{st}$ dominates the path $\mathcal{P}_{st}$, which contradicts the Pareto minimality of the path $\mathcal{P}_{st}$. For maximization the proof is similar. □

Note that Theorem 1 also holds for graphs with negative weights and that an additive objective function with nonnegative edge costs c^k_{ij} is a special case of the strictly increasing function.

4.5.1 Objectives Based on Additive Weight Functions

We will start by showing the case of an objective based on a finite number of additive weight functions. The problem for non-additive objectives is that the value of $C(\mathcal{P}_{it})$ can vary depending on the path taken from s to i. This is also the case when the non-additive objective is based on additive weight functions.

Only the single objective case is considered, although the results easily can be generalized to the multi-objective case by using the definition of Pareto optimality.

Theorem 2 (Gradient domination)**.** *Given a weighted directed graph $G = (V, E)$ let $w_i : E \to \mathbb{R}, i \in \{1, \ldots, n\}$ be additive weight functions on G. Let there be an objective function of the form: $C(\mathcal{P}) = \sum_{i=1}^{n} f_i(w_i(\mathcal{P}))$. Let $\mathcal{P}_{st}$ be composed of subpaths $\mathcal{P}_{sj}$ and $\mathcal{P}_{jt}$ and let $\mathcal{P}'_{st}$ be composed of $\mathcal{P}'_{sj}$ and $\mathcal{P}_{jt}$ (see Figure 4.1). Let*

$$\mathcal{M}_i^- \leq \frac{f_i(w_i(\mathcal{P}'_{st})) - f_i(w_i(\mathcal{P}_{st}))}{w_i(\mathcal{P}'_{st}) - w_i(\mathcal{P}_{st})} \leq \mathcal{M}_i^+, \quad i \in \{1, \ldots, n\},\ w_i(\mathcal{P}_{st}) \neq w_i(\mathcal{P}'_{st}) \tag{4.7}$$

where $\mathcal{M}_i^-, \mathcal{M}_i^+ \in \mathbb{R}$. Moreover let:

$$\sum_{i=1}^{n} \mathcal{M}_i w_i(\mathcal{P}'_{sj}) < \sum_{i=1}^{n} \mathcal{M}_i w_i(\mathcal{P}_{sj}) \tag{4.8}$$

with strict inequality for at least one i. where

$$\mathcal{M}_i = \begin{cases} \mathcal{M}_i^+ & \text{if } w_i(\mathcal{P}'_{sj}) > w_i(\mathcal{P}_{sj}) \\ 1 & \text{if } w_i(\mathcal{P}'_{sj}) = w_i(\mathcal{P}_{sj}) \\ \mathcal{M}_i^- & \text{if } w_i(\mathcal{P}'_{sj}) < w_i(\mathcal{P}_{sj}) \end{cases}$$

Then $C(\mathcal{P}'_{st}) < C(\mathcal{P}_{st})$.

Proof. Assume that (4.8) holds. For each i where $w_i(\mathcal{P}_{sj}) \neq w_i(\mathcal{P}'_{sj})$ we have:

$$\frac{f_i(w_i(\mathcal{P}'_{st})) - f_i(w_i(\mathcal{P}_{st}))}{w_i(\mathcal{P}'_{sj}) - w_i(\mathcal{P}_{sj})} = \frac{f_i(w_i(\mathcal{P}'_{st})) - f_i(w_i(\mathcal{P}_{st}))}{w_i(\mathcal{P}_{jt}) + w_i(\mathcal{P}'_{sj}) - (w_i(\mathcal{P}_{sj}) + w_i(\mathcal{P}_{jt}))} = \frac{f_i(w_i(\mathcal{P}'_{st})) - f_i(w_i(\mathcal{P}_{st}))}{w_i(\mathcal{P}'_{st}) - w_i(\mathcal{P}_{st})}$$

which is less than or equal to $\mathcal{M}_i^+$ if $w_i(\mathcal{P}'_{sj}) > w_i(\mathcal{P}_{sj})$ and which is greater than or equal to $\mathcal{M}_i^-$ if $w_i(\mathcal{P}'_{sj}) < w_i(\mathcal{P}_{sj})$. This implies that

$$f_i(w_i(\mathcal{P}'_{st})) - f_i(w_i(\mathcal{P}_{st})) \leq \mathcal{M}_i(w_i(\mathcal{P}'_{st}) - w_i(\mathcal{P}_{st})) \tag{4.9}$$

It is easy to see that inequality (4.9) also will hold if $w_i(\mathcal{P}_{sj}) = w_i(\mathcal{P}'_{sj})$. This means that

$$\sum_{i=1}^{n} (f_i(w_i(\mathcal{P}'_{st})) - f_i(w_i(\mathcal{P}_{st}))) \leq \sum_{i=1}^{n} (\mathcal{M}_i(w_i(\mathcal{P}'_{st}) - w_i(\mathcal{P}_{st})))$$

and hence

$$\sum_{i=1}^{n} (f_i(w_i(\mathcal{P}'_{st})) \leq \sum_{i=1}^{n} (\mathcal{M}_i(w_i(\mathcal{P}'_{st}) - w_i(\mathcal{P}_{st}))) + \sum_{i=1}^{n} f_i(w_i(\mathcal{P}_{st})) \tag{4.10}$$

Now, adding inequalities (4.8) and (4.10) we get:

$$\sum_{i=1}^{n} (f_i(w_i(\mathcal{P}'_{st})) < \sum_{i=1}^{n} (f_i(w_i(\mathcal{P}_{st}))$$

and hence $C(\mathcal{P}'_{st}) < C(\mathcal{P}_{st})$ which proves the theorem. □

In the next corollary we consider problems with additive weight functions where an objective function is based on two additive criteria. This case is relevant for some of the real life problems considered in Section 4.6.

Corollary 1 (minimization of objective)**.** *Given a weighted directed graph $G = (V, E)$ let $w_1 : E \to \mathbb{R}$ and $w_2 : E \to \mathbb{R}$ be two additive weight functions on G. Let there be an objective function of the form: $C(\mathcal{P}) = w_1(\mathcal{P}) + f(w_2(\mathcal{P}))$ which is to be minimized. Let $\mathcal{P}_{st}$ be composed of subpaths $\mathcal{P}_{sj}$ and $\mathcal{P}_{jt}$ and let $\mathcal{P}'_{st}$ be composed of $\mathcal{P}'_{sj}$ and $\mathcal{P}_{jt}$ (see Figure 4.1). Let*

$$\mathcal{M}^- \leq \frac{f(w_2(\mathcal{P}'_{st})) - f(w_2(\mathcal{P}_{st}))}{w_2(\mathcal{P}'_{st}) - w_2(\mathcal{P}_{st})} \leq \mathcal{M}^+,\ w_2(\mathcal{P}'_{st}) \neq w_2(\mathcal{P}_{st}) \tag{4.11}$$

where $\mathcal{M}^-, \mathcal{M}^+ \in \mathbb{R}$. Moreover let:

$$\begin{array}{lll} w_1(\mathcal{P}'_{sj}) + \mathcal{M}^+ w_2(\mathcal{P}'_{sj}) < w_1(\mathcal{P}_{sj}) + \mathcal{M}^+ w_2(\mathcal{P}_{sj}) & \text{if} & w_2(\mathcal{P}'_{sj}) > w_2(\mathcal{P}_{sj}) \\ w_1(\mathcal{P}'_{sj}) + \mathcal{M}^- w_2(\mathcal{P}'_{sj}) < w_1(\mathcal{P}_{sj}) + \mathcal{M}^- w_2(\mathcal{P}_{sj}) & \text{if} & w_2(\mathcal{P}'_{sj}) < w_2(\mathcal{P}_{sj}) \\ w_1(\mathcal{P}'_{sj}) < w_1(\mathcal{P}_{sj}) & \text{if} & w_2(\mathcal{P}'_{sj}) = w_2(\mathcal{P}_{sj}) \end{array} \tag{4.12}$$

Then $C(\mathcal{P}'_{st}) < C(\mathcal{P}_{st})$.

Corollary 1 states that if two subpaths $\mathcal{P}_{sj}$ and $\mathcal{P}'_{sj}$ ending at the same node j satisfy inequality (4.12), then for minimization problems the path $\mathcal{P}'_{sj}$ dominates $\mathcal{P}_{sj}$ and the latter may be deleted.

Proof. This is the special case of Theorem 2 where $n = 2$ and f_1 is the identity function. □

The dominance rule (4.12) was defined for a minimization problem. In the case of *maximization* Corollary 1 is changed to

Corollary 2 (maximization of objective)**.** *Given a weighted directed graph $G = (V, E)$ let $w_1 : E \to \mathbb{R}$ and $w_2 : E \to \mathbb{R}$ be two additive weight functions on G. Let there be an objective function of the form: $C(p) = w_1(p) + f(w_2(p))$ which is to be maximized. Let $\mathcal{P}_{st}$ be composed of subpaths $\mathcal{P}_{sj}$ and $\mathcal{P}_{jt}$ and let $\mathcal{P}'_{st}$ be composed of $\mathcal{P}'_{sj}$ and $\mathcal{P}_{jt}$ (See Figure 4.1) . Let*

$$\mathcal{M}^- \leq \frac{f(w_2(\mathcal{P}'_{st})) - f(w_2(\mathcal{P}_{st}))}{w_2(\mathcal{P}'_{st}) - w_2(\mathcal{P}_{st})} \leq \mathcal{M}^+,\ w_2(\mathcal{P}'_{st}) \neq w_2(\mathcal{P}_{st}) \tag{4.13}$$

where $\mathcal{M}^-, \mathcal{M}^+ \in \mathbb{R}$. Moreover let:

$$\begin{array}{lll} w_1(\mathcal{P}'_{sj}) + \mathcal{M}^+ w_2(\mathcal{P}'_{sj}) > w_1(\mathcal{P}_{sj}) + \mathcal{M}^+ w_2(\mathcal{P}_{sj}) & \text{if} & w_2(\mathcal{P}'_{sj}) < w_2(\mathcal{P}_{sj}) \\ w_1(\mathcal{P}'_{sj}) + \mathcal{M}^- w_2(\mathcal{P}'_{sj}) > w_1(\mathcal{P}_{sj}) + \mathcal{M}^- w_2(\mathcal{P}_{sj}) & \text{if} & w_2(\mathcal{P}'_{sj}) > w_2(\mathcal{P}_{sj}) \\ w_1(\mathcal{P}'_{sj}) > w_1(\mathcal{P}_{sj}) & \text{if} & w_2(\mathcal{P}'_{sj}) = w_2(\mathcal{P}_{sj}) \end{array} \tag{4.14}$$

Then $C(\mathcal{P}'_{st}) > C(\mathcal{P}_{st})$.

In finding $\mathcal{M}^-$ and $\mathcal{M}^+$ the goal is to maximize $\mathcal{M}^-$ and minimize $\mathcal{M}^+$ so that the number of paths kept for investigation is minimized.

Remark 1. When f is **differentiable** and there exists $\mathcal{M}^-$ and $\mathcal{M}^+$ such that $\mathcal{M}^- \leq f'(x) \leq \mathcal{M}^+$ for all x in the domain of f, then by the Mean Value Theorem we have

$$\mathcal{M}^- \leq \frac{f(w_2(\mathcal{P}'_{st})) - f(w_2(\mathcal{P}_{st}))}{w_2(\mathcal{P}'_{st}) - w_2(\mathcal{P}_{st})} \leq \mathcal{M}^+.$$

Note that on a fixed weight graph G the domain of f can be restricted to a closed interval $[a, b]$ such that $f(w_2(P)) \in [a, b]$ for every simple path P in G. Moreover there could be a lower bound on how much $w_2(\mathcal{P}'_{st}) - w_2(\mathcal{P}_{st})$ can be for two different paths. This could for example be the smallest cost of an edge in G.

Such upper bound on $f(w_2(\mathcal{P}'_{st})) - f(w_2(\mathcal{P}_{st}))$ and lower bound $w_2(\mathcal{P}'_{st}) - w_2(\mathcal{P}_{st})$ could be used to find a possible value for $\mathcal{M}^+$. In the same way possible values for $\mathcal{M}^-$ can be found. Clearly if f is a differentiable bounded function then $\mathcal{M}^+$ can be the maximum of the derivative and $\mathcal{M}^-$ the minimum of the derivative in the bounded region.

Remark 2. When f is **convex** and $w_2 : E \to \mathbb{R}_0^+$ then

$$\frac{f(w_2(\mathcal{P}_{si})) - f(w_2(\mathcal{P}'_{si}))}{w_2(\mathcal{P}_{si}) - w_2(\mathcal{P}'_{si})} \leq \frac{f(w_2(\mathcal{P}_{st})) - f(w_2(\mathcal{P}'_{st}))}{w_2(\mathcal{P}_{st}) - w_2(\mathcal{P}'_{st})} = \frac{f(w_2(\mathcal{P}'_{st})) - f(w_2(\mathcal{P}_{st}))}{w_2(\mathcal{P}'_{st}) - w_2(\mathcal{P}_{st})} \tag{4.15}$$

and thus $\mathcal{M}^-$ can be chosen as:

$$\mathcal{M}^- = \frac{f(w_2(\mathcal{P}_{si})) - f(w_2(\mathcal{P}'_{si}))}{w_2(\mathcal{P}_{si}) - w_2(\mathcal{P}'_{si})} \tag{4.16}$$

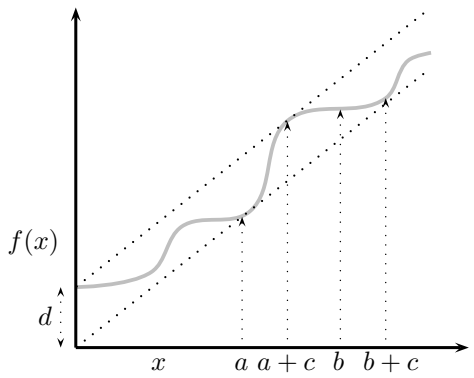


Figure 4.2: The property of f to facilitate the use of Floor domination

when substituting the value of $\mathcal{M}^-$ into the portion of the dominance definition in Corollary 1 containing $\mathcal{M}^-$ (both in the minimization and maximization versions) then regular dominance (principle of optimality) is achieved in that portion.

Moreover if $w_2(\mathcal{P}_{it}) \leq b$ for all simple paths $\mathcal{P}_{it}$ from i to t in G then we can choose $\mathcal{M}^+$ as follows:

$$\mathcal{M}^+ = \frac{f(b + w_2(\mathcal{P}'_{si})) - f(b + w_2(\mathcal{P}_{si}))}{w_2(\mathcal{P}'_{si}) - w_2(\mathcal{P}_{si})} \geq \frac{f(w_2(\mathcal{P}'_{st})) - f(w_2(\mathcal{P}_{st}))}{w_2(\mathcal{P}'_{st}) - w_2(\mathcal{P}_{st})}$$

Similarly when $w_2 : E \to \mathbb{R}_0^-$ then $\mathcal{M}^+$ can be chosen as

$$\mathcal{M}^+ = \frac{f(w_2(\mathcal{P}'_{si})) - f(w_2(\mathcal{P}_{si}))}{w_2(\mathcal{P}'_{si}) - w_2(\mathcal{P}_{si})} \geq \frac{f(w_2(\mathcal{P}'_{st})) - f(w_2(\mathcal{P}_{st}))}{w_2(\mathcal{P}'_{st}) - w_2(\mathcal{P}_{st})}, \tag{4.17}$$

which in the portion containing $\mathcal{M}^+$ reduces to regular dominance in the dominance definition of Corollary 1. If $\exists a \leq w_2(\mathcal{P}_{it})$ for all simple paths $\mathcal{P}_{it}$ from i to t in G then $\mathcal{M}^-$ can be chosen as

$$\mathcal{M}^- = \frac{f(a + w_2(\mathcal{P}_{si})) - f(a + w_2(\mathcal{P}'_{si}))}{w_2(\mathcal{P}_{si}) - w_2(\mathcal{P}'_{si})} \leq \frac{f(w_2(\mathcal{P}_{st})) - f(w_2(\mathcal{P}'_{st}))}{w_2(\mathcal{P}_{st}) - w_2(\mathcal{P}'_{st})}. \tag{4.18}$$

Remark 3. When f is **concave**, $-f$ is convex and therefore we can apply Remark 2 to $-f$ to find suitable values of $\mathcal{M}^-$ and $\mathcal{M}^+$ for f.

These remarks will be used when returning to the objective functions described in Section 4.2.

Another function used in the objectives described in Section 4.6 is the *floor* function. Clearly the *floor* function fulfills the requirements of Corollary 1, however, $\mathcal{M}^+$ is infinite. Using the *floor* dominance stated by Jepsen et al. in [16], we present a general form of domination.

Theorem 3 (Floor domination)**.** *Given a weighted directed graph $G = (V, E)$ let $w_1 : E \to \mathbb{R}$ and $w_2 : E \to \mathbb{R}$ be two additive weight functions on G. Let there be an objective function of the form: $C(\mathcal{P}) = w_1(\mathcal{P}) + f(w_2(\mathcal{P}))$. Let the function f: $\mathbb{R} \to \mathbb{R}$ satisfy $f(a+c) - f(a) - d \leq f(b+c) - f(b)$ for all a,b,c in the domain of f (see Figure 4.2) where a, b satisfy some property D and $d \in \mathbb{R}_0^+$. Let $\mathcal{P}_{st}$ be composed of subpaths $\mathcal{P}_{sj}$ and $\mathcal{P}_{jt}$ and let $\mathcal{P}'_{st}$ be composed of $\mathcal{P}'_{sj}$ and $\mathcal{P}_{jt}$ (see Figure 4.1). Then:*

$$C(\mathcal{P}'_{sj}) + d \leq C(\mathcal{P}_{sj}) \Rightarrow C(\mathcal{P}'_{st}) \leq C(\mathcal{P}_{st}) \tag{4.19}$$

when $w_2(\mathcal{P}'_{sj}), w_2(\mathcal{P}_{sj})$ are satisfying property D.

Proof. Assume $w_1(\mathcal{P}'_{sj}) + f(w_2(\mathcal{P}'_{sj})) + d \leq w_1(\mathcal{P}_{sj}) + f(w_2(\mathcal{P}_{sj}))$ and that $w_2(\mathcal{P}'_{sj}), w_2(\mathcal{P}_{sj})$ satisfy a property D. Then by the additivity of w_1 we have that:

$$w_1(\mathcal{P}'_{st}) + f(w_2(\mathcal{P}'_{sj})) + d \leq w_1(\mathcal{P}_{st}) + f(w_2(\mathcal{P}_{sj})) \tag{4.20}$$

By the property of f we have:

$$f(w_2(\mathcal{P}'_{sj}) + w_2(\mathcal{P}_{jt})) - f(w_2(\mathcal{P}'_{sj})) - d \leq f(w_2(\mathcal{P}_{sj}) + w_2(\mathcal{P}_{jt})) - f(w_2(\mathcal{P}_{sj})) \tag{4.21}$$

by adding the previous two inequalities we get:

$$w_1(\mathcal{P}'_{st}) + f(w_2(\mathcal{P}'_{st})) \leq w_1(\mathcal{P}_{st}) + f(w_2(\mathcal{P}_{st})) \tag{4.22}$$

Theorem 3 is applied to real life problems in Sections 4.6.6 and 4.6.7. See Sections 4.6.6 and 4.6.7 for examples of the determination of d and property D in a real life problem.

4.5.2 Objectives Based on the Max and Min Function

In this subsection we consider a non-monotone objective function defined as the maximum of a set of edge weights visited on the path $\mathcal{P}$. Theorem 4 and its proof is a generalization of the work in [2].

Theorem 4 (Max domination)**.** *Given a weighted directed graph $G = (V, E)$ let $w_1 : E \to \mathbb{R}$ be an additive weight function and $a_j : E \to \mathbb{R}, \quad j \in \{1, ..., n\}$ be a map from the edges of G into $\mathbb{R}$ and let $e \in E$ be an edge in G. Then let $w_2(\mathcal{P}) = \sum_{j=1}^{n} \max_{e\in\mathcal{P}} a_j(e)$. Let there be an objective function of the form $C(p) = w_1(p) + w_2(p)$ to be minimized. Let $\mathcal{P}_{st}$ be composed of subpaths $\mathcal{P}_{si}$ and $\mathcal{P}_{it}$ and let $\mathcal{P}'_{st}$ be composed of $\mathcal{P}'_{si}$ and $\mathcal{P}_{it}$. Moreover let:*

$$w_1(\mathcal{P}'_{si}) + \sum_{j=1}^{n} \max_{e\in\mathcal{P}'_{si}} \{a_j(e)\} < w_1(\mathcal{P}_{si}) + \sum_{j=1}^{n} \max_{e\in\mathcal{P}_{si}} \{a_j(e)\} - F(\mathcal{P}_{si}, \mathcal{P}'_{si}) \tag{4.23}$$

where

$$F(\mathcal{P}, \mathcal{P}') = \sum_{j=1}^{n} \max\{0, \max_{e\in\mathcal{P}} a_j(e) - \max_{e\in\mathcal{P}'} a_j(e)\}.$$

Then $C(\mathcal{P}'_{st}) < C(\mathcal{P}_{st})$.

In other words if two subpaths $\mathcal{P}_{si}$ and $\mathcal{P}'_{si}$ end at the same node i and (4.23) is satisfied, then the label corresponding to subpath $\mathcal{P}'_{si}$ dominates the label corresponding to subpath $\mathcal{P}_{si}$ and hence the latter may be deleted.

Proof. With some trivial case studies it is easy to see that for each $j = 1, \ldots, n$ we have

$$\begin{aligned}\max\left\{0, \max_{e\in\mathcal{P}_{it}}\{a_j(e)\} - \max_{e\in\mathcal{P}'_{si}}\{a_j(e)\}\right\} - \max\left\{0, \max_{e\in\mathcal{P}_{it}}\{a_j(e)\} - \max_{e\in\mathcal{P}_{si}}\{a_j(e)\}\right\} \\ \leq \max\left\{0, \max_{e\in\mathcal{P}_{si}}\{a_j(e)\} - \max_{e\in\mathcal{P}'_{si}}\{a_j(e)\}\right\}\end{aligned}$$

Adding these inequalities for $j = 1, \ldots, n$ together, we obtain

$$\begin{aligned}\sum_{j=1}^{n} \max\left\{0, \max_{e\in\mathcal{P}_{it}}\{a_j(e)\} - \max_{e\in\mathcal{P}'_{si}}\{a_j(e)\}\right\} - \sum_{j=1}^{n} \max\left\{0, \max_{e\in\mathcal{P}_{it}}\{a_j(e)\} - \max_{e\in\mathcal{P}_{si}}\{a_j(e)\}\right\} \\ \leq \sum_{j=1}^{n} \max\left\{0, \max_{e\in\mathcal{P}_{si}}\{a_j(e)\} - \max_{e\in\mathcal{P}'_{si}}\{a_j(e)\}\right\}\end{aligned}$$

By the definition of $F(\mathcal{P}, \mathcal{P}')$, the above inequality is the same as:

$$\begin{aligned}\sum_{j=1}^{n} \max\left\{0, \max_{e\in\mathcal{P}_{it}}\{a_j(e)\} - \max_{e\in\mathcal{P}'_{si}}\{a_j(e)\}\right\} \\ \leq \sum_{j=1}^{n} \max\left\{0, \max_{e\in\mathcal{P}_{it}}\{a_j(e)\} - \max_{e\in\mathcal{P}_{si}}\{a_j(e)\}\right\} + F(\mathcal{P}_{si}, \mathcal{P}'_{si})\end{aligned} \tag{4.24}$$

Adding inequality (4.24) to the assumption (4.23) we achieve:

$$w_1(\mathcal{P}'_{si}) + \sum_{j=1}^{n} \max_{e\in\mathcal{P}'_{si}}\{a_j(e)\} + \sum_{j=1}^{n} \max\left\{0, \max_{e\in\mathcal{P}_{it}}\{a_j(e)\} - \max_{e\in\mathcal{P}'_{si}}\{a_j(e)\}\right\}$$
$$< w_1(\mathcal{P}_{si}) + \sum_{j=1}^{n} \max_{e\in\mathcal{P}_{si}}\{a_j(e)\} + \sum_{j=1}^{n} \max\left\{0, \max_{e\in\mathcal{P}_{it}}\{a_j(e)\} - \max_{e\in\mathcal{P}_{si}}\{a_j(e)\}\right\}$$

It is easy to check that this is the same as:

$$w_1(\mathcal{P}'_{si}) + \sum_{j=1}^{n} \max\left\{\max_{e\in\mathcal{P}_{it}}\{a_j(e)\}, \max_{e\in\mathcal{P}'_{si}}\{a_j(e)\}\right\} < w_1(\mathcal{P}_{si}) + \sum_{i=1}^{n} \max\left\{\max_{e\in\mathcal{P}_{it}}\{a_j(e)\}, \max_{e\in\mathcal{P}_{si}}\{a_j(e)\}\right\}$$

which is equivalent to

$$w_1(\mathcal{P}'_{si}) + \sum_{j=1}^{n} \max_{e\in\mathcal{P}'_{st}}\{a_j(e)\} < w_1(\mathcal{P}_{si}) + \sum_{j=1}^{n} \max_{e\in\mathcal{P}_{st}}\{a_j(e)\}$$

By adding $w_1(\mathcal{P}_{it})$ to both sides we get the desired result $C(\mathcal{P}'_{st}) < C(\mathcal{P}_{st})$. □

Theorem 5 (Min domination)**.** *Given a weighted directed graph $G = (V, E)$ let $w_1 : E \to \mathbb{R}$ be an additive weight function and $a_i : E \to \mathbb{R}, \quad i \in \{1, ..., n\}$ be a map from the edges of G to the reals. Then let $w_2(\mathcal{P}) = \sum_{i=1}^{n} \min_{e\in\mathcal{P}} a_i(e)$. Let there be an objective function of the form $C(\mathcal{P}) = w_1(\mathcal{P}) - w_2(\mathcal{P})$ to be minimized. Let $\mathcal{P}_{st}$ be composed of subpaths $\mathcal{P}_{si}$ and $\mathcal{P}_{it}$ and let $\mathcal{P}'_{st}$ be composed of $\mathcal{P}'_{si}$ and $\mathcal{P}_{it}$. Let $F(\mathcal{P}, \mathcal{P}') = \sum_{j=1}^{n} \max\{0, \min_{e\in\mathcal{P}} a_j(e) - \min_{e\in\mathcal{P}'} a_j(e)\}$. Moreover let:*

$$C(\mathcal{P}'_{si}) < C(\mathcal{P}_{si}) - F(\mathcal{P}_{si}, \mathcal{P}'_{si}) \tag{4.25}$$

Then $C(\mathcal{P}'_{st}) < C(\mathcal{P}_{st})$.

In the cases where domination (4.25) of Theorem 5 holds we can when minimizing $C(\mathcal{P})$ eliminate paths $C(\mathcal{P}_{si})$.

Proof. Clearly, since $\max_{e\in\mathcal{P}}\{a_j(e)\} = -\min_{e\in\mathcal{P}}\{-a_j(e)\}$ and therefore

$$C(\mathcal{P}) = w_1(\mathcal{P}) - \sum_{i=1}^{n} \min_{e\in\mathcal{P}}\{-a_i(e)\} = w_1(\mathcal{P}) + \sum_{i=1}^{n} \max_{e\in\mathcal{P}} a_i(e).$$

By Theorem 4, it follows that Theorem 5 holds. □

In Section 4.6.3 and 4.6.4 the Theorem 4 is applied to real life problems. For an example of how $F(\mathcal{P}, \mathcal{P}')$ is determined see Sections 4.6.3 and 4.6.4.

4.5.3 Goal domination

Goal domination is a well known and commonly used way of eliminating labels (see [19]). This domination does not require monotonicity, however it can only be used for objectives $C(P)$, that are non-decreasing or non-increasing and a feasible solution to the problem must be found before goal domination can be used. The goal domination checks each new label created for whether it is *dominated* by a label at the destination.

To improve the goal domination one can find lower bounds (upper bounds in the case of a maximization problem) on all objectives and pairs of vertices. This can be done by preprocessing the graph data. When using preprocessing, the goal domination can be extended so that the lower bound of the remaining path is added to the label when testing whether it is dominated by labels at the destination.

Dumitrescu and Boland [7] show that preprocessing can reduce computation time significantly on resource constrained shortest path problems. Preprocessing is useful in graphs where the edge weights seldom change. Nevertheless, the storage requirements increase as there can be a quadratic number of lower bounds. However, each lower bound will not be very space-consuming as it is a simple number. Lower bounds generated "on the fly" are often used in graphs where the vertices are placed in a coordinate system and therefore the Euclidean distance between nodes can be calculated "on the fly" [19]. Another lower bound method [19] is to make simplified versions of the graph in which a specific lower bound can be found at a given shortest path request.

4.6 Solving Real Life Shortest Path Problems

With the properties covered in previous section we now return to the objectives (A) through (G) listed in Section 4.2. We describe how the developed framework can be adapted to the considered objectives.

4.6.1 Multiplicative cost function

The multiplicative objective described in (A) is given as follows: Each edge e_{ij} has a probability $\pi_{ij} \in (0, 1]$ for being traversed successfully, and the corresponding objective is

$$C(\mathcal{P}) = \prod_{e_{ij} \in \mathcal{P}} \pi_{ij}$$

The objective is monotone since we have ([23] Example 6.3.2)

$$\max \log C(\mathcal{P}) = \max \log \prod_{e_{ij} \in \mathcal{P}} \pi_{ij} = \max \sum_{e_{ij} \in \mathcal{P}} \log \pi_{ij} \tag{4.26}$$

Defining $w_1 : E \to \mathbb{R}_0^+$ as $w_1(e_{ij}) = \log \pi_{ij}$ and choosing $f = e^x$, Theorem 1 gives that the objective is monotone. Therefore all subpaths are also optimal, and if the problem contains several monotone objective functions all subpaths are Pareto optimal. Notice that $w_1(e_{ij}) \geq 0$ since all $0 < \pi_{ij} \leq 1$.

4.6.2 Combined distance and probability function

The second objective (B) is a combination of the two criteria distance d and probability $\pi \in (0, 1]$. The objective is written as $-d + \lambda\pi$ with $\lambda > 0$ which is to be maximized. Let $f(x) = \lambda a^x$ so that $f(\log_a(\pi(\mathcal{P}))) = \lambda\pi(\mathcal{P})$. Let $w_1(\mathcal{P}) = -d(\mathcal{P})$ and $w_2(\mathcal{P}) = \log_a(\pi(\mathcal{P}))$. Since $\log_a(\pi(\mathcal{P}))$ is additive, we can use Corollary 2. Since $\log_a(\pi(\mathcal{P})) : E \to \mathbb{R}_0^-$ and f is convex, we can use Remark 2. By Remark 2 the only case where the *principle of optimality* does not work is the case where $w_2(\mathcal{P}'_{si}) > w_2(\mathcal{P}_{si})$. Therefore, we need to find a good $\mathcal{M}^-$. We choose $\mathcal{M}^- = (f(b + w_2(\mathcal{P}_{si})) - f(b + w_2(\mathcal{P}'_{si})))/(w_2(\mathcal{P}_{si}) - w_2(\mathcal{P}'_{si}))$ where $b \leq w_2(\mathcal{P}_{it})$ for all simple paths $\mathcal{P}_{it}$ from i to t on G as suggested in Remark 2 Equation (4.18). Inserting this value into the definition of dominance for the case $w_2(\mathcal{P}'_{si}) > w_2(\mathcal{P}_{si})$ in Corollary 2 we get:

$$w_1(\mathcal{P}'_{si}) > w_1(\mathcal{P}_{si}) + f(b + w_2(\mathcal{P}_{si})) - f(b + w_2(\mathcal{P}'_{si})) \tag{4.27}$$

A lower bound L for $\pi(\mathcal{P}_{it})$ where $\mathcal{P}_{it}$ is an arbitrary simple path from i to t on G results in $\log_a(L)$ being a lower bound for $\log_a(\pi(\mathcal{P}_{it}))$. Therefore we can substitute b with $\log_a(L)$. Moreover, we can use that $f(x) = \lambda a^x$, $w_1(\mathcal{P}) = -d(\mathcal{P})$ and $w_2(\mathcal{P}) = \log_a(\pi(\mathcal{P}))$ to rewrite (4.27) as:

$$w_1(\mathcal{P}'_{si}) > w_1(\mathcal{P}_{si}) + \lambda a^{\log_a(L) + \log_a(\pi(\mathcal{P}_{si}))} - \lambda a^{\log_a(L) + \log_a(\pi(\mathcal{P}'_{si}))}$$

this inequality is the same as

$$w_1(\mathcal{P}'_{si}) > w_1(\mathcal{P}_{si}) + \lambda L\pi(\mathcal{P}_{si}) - \lambda L\pi(\mathcal{P}'_{si}),$$

and hence

$$-d(\mathcal{P}'_{si}) > -d(\mathcal{P}_{si}) + \lambda L\pi(\mathcal{P}_{si}) - \lambda L\pi(\mathcal{P}'_{si})). \tag{4.28}$$

This means that the dominance of Corollary 2 in this case is:

If

$$-d(\mathcal{P}'_{si}) + \pi(\mathcal{P}'_{si}) > -d(\mathcal{P}_{si}) + \pi(\mathcal{P}_{si}) \quad \text{and} \quad \pi(\mathcal{P}'_{si}) \leq \pi(\mathcal{P}_{si}) \tag{4.29}$$

$$-d(\mathcal{P}'_{si}) > -d(\mathcal{P}_{si}) + \lambda L\pi(\mathcal{P}_{si}) - \lambda L\pi(\mathcal{P}'_{si})) \quad \text{and} \quad \pi(\mathcal{P}'_{si}) > \pi(\mathcal{P}_{si}) \tag{4.30}$$

then $C(\mathcal{P}'_{st}) > C(\mathcal{P}_{st})$.

Carraway et al. [4] propose the following algorithm for solving the dominance test. Let P be a path from a vertex s to a vertex i then d is the distance of the path P and π is the probability of the path P. Let the path P' be a path from s to i different from P and let $\hat{d}$ and $\hat{\pi}$ be the distance and probability on that path.

Algorithm by Carraway et al.:

Step 0. Designate (d, π) and $(\hat{d}, \hat{\pi})$ such that $d \leq \hat{d}$.

Step 1. If $d = \hat{d}$ and $\pi \leq \hat{\pi}$ then delete (d, π) and stop; else if $d = \hat{d}$ and $\pi > \hat{\pi}$ then delete $(\hat{d}, \hat{\pi})$ and stop.

Step 2. If $\pi = \hat{\pi}$, delete $(\hat{d}, \hat{\pi})$ and stop.

Step 3. If $\hat{d} - d \leq \lambda b_j(\hat{\pi} - \pi)$ where b_j is the minimum of the probability on the remaining path. Delete (d, π) and stop.

Step 4. If $d - \hat{d} \leq \lambda m_j(\pi - \hat{\pi})$ where m_j is the maximum of the probability on the remaining path. Delete $(\hat{d}, \hat{\pi})$ and stop.

Step 5. Retain both returns and stop.

In [4] Step 2 was written as follows: *If $\pi \leq \hat{\pi}$, delete $(\hat{d}, \hat{\pi})$ and stop.* As this does not hold, it is presumed to be a typing error and we have inserted = instead of $\leq$. Note that by using a non-strict inequality, Carraway et al. only get the *minimal complete* set of solutions. Clearly, by making the inequality non-strict, our dominance method, (4.29) and (4.30), will find exactly the *minimal complete* set of Pareto optimal solutions.

We now show that Theorem 2 allows for the elimination of all of the paths eliminated by the algorithm of Carraway et al.

It can easily be shown that all paths deleted by step 1 and 2 are eliminated by Corollary 2, inequalities (4.29) and (4.30).

In order to prove the same for step 3 we have to show that when $d \leq \hat{d}$ and $\hat{d} - d \leq \lambda b_j(\hat{\pi} - \pi)$, where b_j is the minimum of the probabilities, on the remaining path $(\pi(P_{it}))$, then one of the domination statements (4.29) and (4.30) holds. Let (d, π) be the pair $(d(P_{si}), \pi(P_{si}))$ and $(\hat{d}, \hat{\pi})$ be the pair $(d(P'_{si}), \pi(P'_{si}))$. Since $b_j > 0$ we can choose the lower bound $L = b_j > 0$. When $\hat{\pi} < \pi$ no path is deleted in step 3. Now insert the values in the inequality of step 3:

$$d(P'_{si}) - d(P_{si}) \leq \lambda L(\pi(P'_{si}) - \pi(P_{si})) \quad \Rightarrow \quad -d(P'_{si}) + d(P_{si}) \geq \lambda L(-\pi(P'_{si}) + \pi(P_{si})) \tag{4.31}$$

$$\Rightarrow \quad -d(P'_{si}) \geq -d(P_{si}) + \lambda L(\pi(P_{si}) - \pi(P'_{si})) \tag{4.32}$$

which by (4.30) means that Corollary 2 eliminates (d, π) when $\hat{\pi} > \pi$. In the case where $\hat{\pi} = \pi$ it is easy to show that Corollary 2 eliminates (d, π).

To show that a path deleted in step 4 is also deleted by (4.29) and (4.30), set $(d, \pi) = (d(\mathcal{P}'_{si}), \pi(\mathcal{P}'_{si}))$ and $(\hat{d}, \hat{\pi}) = (d(\mathcal{P}_{si}), \pi(\mathcal{P}_{si}))$ and note that in the case where $\pi > \hat{\pi}$, the inequality (4.30) is trivially true. If $\pi \leq \hat{\pi}$, then we must show that the inequality of step 4 implies (4.29). But this must be true as (4.29) amounts to the principle of optimality (regular domination).

4.6.3 Maximum of commissions

The objective described in (C) is taken from [2]. The cost of a path is calculated as the price of the edges (transports) plus some commission to agents.

Let the set of agents be $1, \ldots, A$ and let S_a be the edges covered by agent $a \in \{1, \ldots, A\}$. Each edge is covered by at most one agent. Each edge e_{ij} has a corresponding commission c_{ij}^a. An agent is paid the largest commission he is entitled to on the path $\mathcal{P}$. The objective is then

$$C(\mathcal{P}) = \sum_{e_{ij} \in \mathcal{P}} c_{ij} + \sum_{a=1}^{A} \max_{e_{ij} \in S_a \cap \mathcal{P}} \{0, c_{ij}^a\}$$

In this case we can use Theorem 4 to obtain that

$$\sum_{e_{ij} \in \mathcal{P}'_{sh}} c_{ij} + \sum_{a=1}^{A} \max_{e_{ij} \in S_a \cap \mathcal{P}'_{sh}} \{0, c_{ij}^a\} < \sum_{e_{ij} \in \mathcal{P}_{sh}} c_{ij} + \sum_{a=1}^{A} \max_{e_{ij} \in S_a \cap \mathcal{P}_{sh}} \{0, c_{ij}^a\} - F(\mathcal{P}_{sh}, \mathcal{P}'_{sh})$$

where

$$F(\mathcal{P}_{sh}, \mathcal{P}'_{sh}) = \sum_{a=1}^{A} \max\{0, \max_{e_{ij} \in S_a \cap \mathcal{P}_{sh}} \{0, c_{ij}^a\} - \max_{e_{ij} \in S_a \cap \mathcal{P}'_{sh}} \{0, c_{ij}^a\}\}$$

implies $C(\mathcal{P}') < C(\mathcal{P})$.

4.6.4 Number of zones visited

In the *zone system* objective described in (D) the price is calculated depending entirely on the zones passed on the journey. More formally let the set of edges be divided into zones $Z_1, \ldots, Z_n$. Without loss of generality we may assume that each edge only corresponds to one zone, as we otherwise may split the edge. Let w_e have the value 1 for all edges e.

The objective then is to minimize the number of different zones on a path $\mathcal{P}$. The objective can be written as follows:

$$\min : C(\mathcal{P}) = \sum_{h=1}^{n} \max_{e \in Z_h \cap \mathcal{P}} \{0, w_e\} \tag{4.33}$$

To use Theorem 4 on the zone system each edge must be assigned the zones it travels through. Let

$$F(\mathcal{P}, \mathcal{P}') = \sum_{h=1}^{n} \max\{0, \max_{e \in Z_h \cap \mathcal{P}} \{0, w_e\} - \max_{e \in Z_h \cap \mathcal{P}'} \{0, w_e\}\}$$

By Theorem 4 we get that

$$\sum_{h=1}^{n} \max_{e \in Z_h \cap \mathcal{P}'_{si}} \{0, w_e\} < \sum_{h=1}^{n} \max_{e \in Z_h \cap \mathcal{P}_{si}} \{0, w_e\} - F(\mathcal{P}_{si}, \mathcal{P}'_{si}) \tag{4.34}$$

implies that the $C(\mathcal{P}'_{st}) < C(\mathcal{P}_{st})$.

However for objectives of the form $\sum_{h=1}^{n} \max_{e \in Z_h \cap \mathcal{P}} \{0, w_e\}$ where $w_1(\mathcal{P})$ is 0 for all paths, Theorem 4 will not be able to eliminate any simple subpaths from the investigation. This means that in this case no subpath will be eliminated by the dominance of Theorem 4. In this case *goal domination* must be used to eliminate paths that are not Pareto optimal compared to a solution that is already found. This, however, requires that a solution is found. Another way to circumvent the problem is to combine an additive criterion with the price in the objective so that $w_1(\mathcal{P})$ will have a value greater than zero. The other criterion could for example be time or distance.

4.6.5 Maximum zone distance from origin

The objective (E) of finding the maximum zone distance from a origin can be handled by creating new zones around the source s. Let Z'_1 be the zone in which s is located. Let Z'_2 be the set of zones that are adjacent to Z'_1. In general Z'_h is the set of zones adjacent to Z'_{h-1}. Note that to get to the distance of three zones of the origin one must have passed zones distance two and one from origin. Let the cost function $c_{ij} = c_h$ where $e_{ij} \in Z'_h$ so that h is the number of zones visited. In this case the objective function is:

$$C(\mathcal{P}) = \max_{e_{ij} \in Z' \cap \mathcal{P}} \{c_{ij}\} \tag{4.35}$$

Clearly in this case if

$$\max_{e_{ij} \in Z' \cap \mathcal{P}'_{si}} \{c_{ij}\} \leq \max_{e_{ij} \in Z' \cap \mathcal{P}_{si}} \{c_{ij}\} \tag{4.36}$$

then $C(\mathcal{P}'_{st}) \leq C(\mathcal{P}_{st})$. Notice that it is possible for the domination to eliminate paths if only the *minimal complete* set is desired. If all paths of minimum value are desired then *goal domination* must be used.

4.6.6 Zone distance and time

The objective (F) is another version of objective (E). However this objective takes a time and a number of zones and returns a cost. As in the Copenhagen ticket system, we suppose the cost function is the number of time intervals times a factor σ. Let us suppose that the time intervals all are of some size k. The time is clearly additive and, as in the model of (E), the zones are not. It is a general assumption that the cost function is monotonously increasing. Moreover the time and zone criteria are not independent in the objective function as the cost is the maximum of the cost of the ticket for the time used and the cost of the ticket for the traveled number of zones. The objective now becomes:

$$C(\mathcal{P}) = \max \left\{ c \left(\sum_{e_{ij} \in \mathcal{P}} t_{ij} \right), \max_{e_{ij} \in Z' \cap \mathcal{P}} \{c_{ij}\} \right\} \tag{4.37}$$

From the dominance of objective (E) we have that:

$$\max_{e_{ij} \in Z' \cap \mathcal{P}_{si}} \{c_{ij}\} \leq \max_{e_{ij} \in Z' \cap \mathcal{P}'_{si}} \{c_{ij}\} \Rightarrow \max_{e_{ij} \in Z' \cap \mathcal{P}_{st}} \{c_{ij}\} \leq \max_{e_{ij} \in Z' \cap \mathcal{P}'_{st}} \{c_{ij}\} \tag{4.38}$$

Since the cost function is the product of a factor σ and the number of time intervals, and all time intervals are of same size k, we get

$$c \left(\sum_{e_{ij} \in \mathcal{P}} t_{ij} \right) = \sigma \left\lceil \frac{1}{k} \sum_{e_{ij} \in \mathcal{P}} t_{ij} \right\rceil \tag{4.39}$$

It is easy to see that:

$$\sigma \left\lceil \frac{a+c}{k} \right\rceil - \sigma \left\lceil \frac{a}{k} \right\rceil + \sigma \geq \sigma \left\lceil \frac{b+c}{k} \right\rceil - \sigma \left\lceil \frac{b}{k} \right\rceil \tag{4.40}$$

When $a \bmod k \geq b \bmod k$ and for all other cases regular domination holds.

For two paths $\mathcal{P}$ and $\mathcal{P}'$, we now by Theorem 3 have that:

$$\sigma \left\lceil \frac{1}{k} \sum_{e_{ij} \in \mathcal{P}'_{si}} t_{ij} \right\rceil - \sigma \geq \sigma \left\lceil \frac{1}{k} \sum_{e_{ij} \in \mathcal{P}_{si}} t_{ij} \right\rceil \Rightarrow \sigma \left\lceil \frac{1}{k} \sum_{e_{ij} \in \mathcal{P}'_{st}} t_{ij} \right\rceil \geq \sigma \left\lceil \frac{1}{k} \sum_{e_{ij} \in \mathcal{P}_{st}} t_{ij} \right\rceil \tag{4.41}$$

holds when $\sum_{e_{ij} \in \mathcal{P}'_{si}} t_{ij} \bmod k \geq \sum_{e_{ij} \in \mathcal{P}_{si}} t_{ij} \bmod k$. Clearly if:

$$\sigma\left\lceil \frac{1}{k}\sum_{e_{ij}\in\mathcal{P}_{si}} t_{ij}\right\rceil \leq \begin{cases} \sigma\left\lceil \frac{1}{k}\sum_{e_{ij}\in\mathcal{P}'_{si}} t_{ij}\right\rceil - \sigma & \text{for } \sum_{e_{ij}\in\mathcal{P}'_{si}} t_{ij} \bmod k \geq \sum_{e_{ij}\in\mathcal{P}_{si}} t_{ij} \bmod k \\ \sigma\left\lceil \frac{1}{k}\sum_{e_{ij}\in\mathcal{P}'_{si}} t_{ij}\right\rceil & \text{otherwise} \end{cases}$$

and

$$\max_{e_{ij}\in Z'\cap\mathcal{P}_{si}}\{c_{ij}\} \leq \max_{e_{ij}\in Z'\cap\mathcal{P}'_{si}}\{c_{ij}\} \tag{4.42}$$

then $C(\mathcal{P}_{st}) \leq C(\mathcal{P}'_{st})$.

4.6.7 Modulo k penalties

Let S be a given set of edges. We consider a shortest path problem where we pay an additional penalty $\sigma \geq 0$ each time the set S has been visited k times. Our objective is hence

$$C(\mathcal{P}) = \sum_{e_{ij}\in\mathcal{P}} w_{ij} + \sigma\left\lfloor \frac{1}{k}\sum_{e_{ij}\in S\cap\mathcal{P}} c_{ij}\right\rfloor = W(\mathcal{P}) + \sigma\left\lfloor \frac{1}{k}C_S(\mathcal{P})\right\rfloor \tag{4.43}$$

Jepsen et al. [16] handle this problem by using a dominance criterion that takes into account when the cost $\sigma\lfloor c_{ij}/k\rfloor$ is to be paid. In other words, if two subpaths $\mathcal{P}_{si}$ and $\mathcal{P}'_{si}$ end at the same node i and either

$$C(\mathcal{P}'_{si}) + \sigma \leq C(\mathcal{P}_{si}) \quad \text{and} \quad C_S(\mathcal{P}'_{si}) \bmod k \geq C_S(\mathcal{P}_{si}) \bmod k \tag{4.44}$$

$$C(\mathcal{P}'_{si}) \leq C(\mathcal{P}_{si}) \quad \text{and} \quad C_S(\mathcal{P}'_{si}) \bmod k \leq C_S(\mathcal{P}_{si}) \bmod k \tag{4.45}$$

then $\mathcal{P}'_{si}$ dominates $\mathcal{P}_{si}$.

We have previously seen that:

$$\sigma\left\lfloor \frac{a+c}{k}\right\rfloor - \sigma\left\lfloor \frac{a}{k}\right\rfloor - \sigma \leq \sigma\left\lfloor \frac{b+c}{k}\right\rfloor - \sigma\left\lfloor \frac{b}{k}\right\rfloor \tag{4.46}$$

holds when $a \bmod k \geq b \bmod k$ and for all other cases regular domination holds.

By Theorem 3 we now have that:

$$\begin{aligned} W(\mathcal{P}_{si}) + \sigma\left\lfloor \frac{1}{k}C_S(\mathcal{P}_{si})\right\rfloor + \sigma &\leq \sigma\left\lfloor \frac{1}{k}W(\mathcal{P}'_{si}) + C_S(\mathcal{P}'_{si})\right\rfloor \\ \Rightarrow W(\mathcal{P}_{st}) + \sigma\left\lfloor \frac{1}{k}C_S(\mathcal{P}_{st})\right\rfloor &\leq W(\mathcal{P}'_{st}) + \sigma\left\lfloor \frac{1}{k}C_S(\mathcal{P}'_{st})\right\rfloor \end{aligned} \tag{4.47}$$

holds when $C_S(\mathcal{P}_{si}) \bmod k \geq C_S(\mathcal{P}'_{si}) \bmod k$ and otherwise regular domination holds. Since Theorem 3 was inspired by the domination found in Jepsen et al. [16] it is not a surprise that it holds for their case.

4.7 Computational Results

In this section we show that even though non-additive criteria functions are hard to solve in theory, several real-life problems are tractable in practice. The label-correcting algorithm was implemented in C++ and all tests were carried out on a 2.1 GHz Pentium processor.

The considered instances are based on real-life data from a shipping company which wants to find the Pareto optimal paths for transporting a container from s to t when considering various objectives. The given network data contained 15 vertices and 125 edges. In order to construct larger instances, the network has been upscaled.

In all tests we have restricted the number of transfers to at most 10. This number is quite large and in reality could be set lower. For comparison, we have run tests where the Pareto optimal solutions are found without using the nonadditive domination. In some of the graph instances the tests not using nonadditive domination while using goal dominance only were omitted from the test set as they were too time consuming.

To improve the performance of the algorithm we have applied some extra methods for eliminating undesirable paths such as *goal dominance* described in Section 4.5.3. In the case of *goal dominance* the standard elimination is used for all objectives. Another way to improve elimination is to preprocess the data by creating lower bounds between all pairs of vertices for all objectives. This method is described in Section 4.5.3. The lower bounds can be used to see if a subpath will be sure to yield a *dominated* path. This process uses the subpath values combined with the lower bounds of all objectives from the current vertex to the destination to check for dominance against a path already found.

Transfer, time and price with maximum of commissions objectives									
Instance	Number of Pareto optimal solutions found on 100 random requests			Time for preprocessing (sec)		Number of hubs		Number of departures	
R1	571			3.96		91		1176	
R2	491			137.24		221		4176	
R3	583			2934.72		321		10176	
With weighted domination									
	No optimization			**Goal dominance**			**Lower bounds**		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
Average time (sec)	0.24	0.94	4.297	0.12	0.40	1.98	0.08	0.23	1.05
Longest time (sec)	0.54	1.72	9.23	0.39	1.30	7.11	0.25	0.91	4.84
Average # of labels	158566	688883	2.36368e+06	80255	301302	1.21166e+06	51399	158637	532142
Without weighted domination									
	Goal dominance			**Lower bounds**					
	R1	R2	R3	R1	R2	R3			
Average time (sec)	15.74	-	-	3.71	3.87	7.05			
Longest time (sec)	543.04	-	-	243.25	141.82	147.18			
Average # of labels	416319	-	-	165849	458275	1.12563e+06			

Table 4.1: This table shows the running times and the number of labels generated when finding the Pareto optimal paths in a graph. The tests are done with 100 random requests where one of the objectives contains a max function.

The tests reported in Table 4.1, considers a real-life shipping problem similar to case C described in Section 4.2. The shipping company wants to find the Pareto optimal paths for transporting a container from s to t when considering the time, the number of transfers, and the cost. The cost is non-additive as it includes payments to agents along the path. Even though the same agent may be responsible for several vertices on a path, the agent only gets paid once, corresponding to the largest commission the agent is entitled to along the path taken. To solve this problem we have used *max-dominance* (Theorem 4) for the cost objective and regular *dominance* (Theorem 1) for the time and transfer objectives.

To illustrate the strength of the domination presented in this paper we have compared the results with results where we only eliminate a path at an intermediate point if all the commissions

to agents on the eliminated path are smaller than or equal to the commissions paid on the dominating path. Note that the random request are different for the different test instances. The fact that the request are different for the instances may explain the surprising result that the longest time it takes to solve a problem without using *max-dominance* takes longer for the smaller $R1$ instance than for the other two larger instances.

Transfer, time, price with maximum of commissions and probability objectives

Instance	Number of Pareto optimal solutions found on 100 random requests	Time for preprocessing (sec)	Number of hubs	Number of departures
R1	652	0.24	21	176
R2	3665	9.9	91	1176
R3	3242	55.15	121	2176

With weighted domination

	No optimization			**Goal dominance**			**Lower bounds**		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
Average time (sec)	0.06	7.18	15.60	0.04	2.19	6.66	0.02	0.46	1.01
Longest time (sec)	0.18	32.83	70.04	0.18	19.17	61.33	0.08	4.77	14.26
Average # of labels	36978	992223	2.37416e+06	22616	439291	1.12678e+06	10122	124312	261952

Without weighted domination

	Goal dominance			**Lower bounds**		
	R1	R2	R3	R1	R2	R3
Average time (sec)	3.21	-	-	0.07	34.41	55.02
Longest time (sec)	130.21	-	-	1.08	2244.09	2174.12
Average # of labels	143011	-	-	22690	454276	9003371

Table 4.2: This table shows the running times and the number of labels generated when finding the Pareto optimal paths in a graph. The tests are done with 100 random requests where one of the objectives is the probability function.

In the second test, reported in Table 4.2, the four objectives of time, transfers, cost and the probability of reaching destination are considered. The probability objective is described in case A of Section 4.2. As we did not have access to real-life probability data, the probability of each edge was uniformly randomly distributed. It is clear that with an additional objective the complexity of the problem will increase. This can be seen from the fact that the average number of Pareto optimal paths in Table 4.2 for instance $R2$ and $R3$ is more than 32 per request, where real-life instances typically have fewer. Note that the probability objective uses regular domination as it has the monotonicity property. However the price objective is the nonadditive objective from case C described in Section 4.2. To solve this problem we have used *max-dominance* (Theorem 4) for the cost objective and regular *dominance* (Theorem 1) for the time, transfers and probability objectives. In the comparison without using nonadditive domination we have changed the nonadditive domination on the price objective as in Table 4.1. It is clear that the *max-dominance* has a significant impact on the running time as does the lowerbound method even though the the time used on preprocessing increases significantly with the instance size.

In the third test, reported in Table 4.3, the three objectives time, price and the weighted objective of transfers and probability was considered. The weighted function of an additive function and the probability function is described for the distance and probability in case B of Section 4.2. The price objective is the nonadditive objective from case C described in Section 4.2. To solve this problem we have used Corollary 2 for the weighted transfer and probability objective and regular *dominance* (Theorem 1) for the time objective and the *max-dominance* (Theorem 4) for the price objective.

For comparison tests reported in Table 4.1, 4.2 and 4.3 are also run without using the domina-

Time, price with maximum of commissions and transfer-probability objectives			
Instance	Number of Pareto optimal solutions found on 100 random requests	Number of hubs	Number of departures
R1	408	21	176
R2	700	91	1176
R3	799	121	2176

With weighted domination									
	No optimization			**Goal dominance**			**Lower bounds**		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
Average time (sec)	0.04	0.37	1.40	0.02	0.16	0.55	0.01	0.10	0.34
Longest time (sec)	0.11	0.84	9.44	0.08	0.69	4.06	0.08	0.38	2.33
Average # of labels	20067	209156	477149	11536	99848	210950	7860	58740	125878

Without weighted domination						
	Goal dominance			**Lower bounds**		
	R1	R2	R3	R1	R2	R3
Average time (sec)	0.81	-	-	0.08	1.76	6.19
Longest time (sec)	28.50	-	-	1.82	76.89	146.86
Average # of labels	54585	-	-	20206	183152	394376

Table 4.3: This table shows the running times and the number of labels generated when finding the Pareto optimal paths in a graph. The tests are done with 100 random requests where one of the objectives is the probability combined with transfer function.

tion of Corollary 2. In that case the weighted objective of transfer and probability is dominated when the transfers are greater and the probability is smaller than those on the dominating path. For the time objective we use ordinary domination, while the price objective makes use of the same domination as in the previous tests.

The results clearly indicate that the presented tightened dominance methods for non-additive objectives significantly decrease the number of labels generated. In fact for the large graphs, less than a third of the labels are generated when using the non-additive domination methods for objectives. The data used for the lower bound method generated during preprocessing can be used for all requests as long as the graph and objectives are the same. The time used on preprocessing clearly depends on the graph size. If the graph seldom is changed, it is computationally cheap to find the lower bounds. In the case of Table 4.2 instance $R3$ it is evident that preprocessing reduces the running time and number of generated labels significantly.

Comparing instances $R2$, $R3$ in the Tables 4.2 and 4.3 it is, not surprisingly, clear that adding objectives has an impact on the complexity of the problem with an significant increase in the number of Pareto optimal solutions. For all the tests, the tightened domination reduces the average running time by at least a factor three and the longest running time by at least a factor two. One can also see that the improvement increases as the graph gets larger.

4.8 Conclusion

This paper has presented some general techniques to restrict the subpaths that need to be investigated in a dynamic programming algorithm, when solving shortest path problems with several non-additive functions. The dynamic programming method for shortest path is simple and easy to adjust when the complicated cost functions that arise in real-life applications are encountered. The domination criteria presented in the theorems can, as shown in Section 4.6, be used on different real-life applications. However it should be noted that there are still many non-additive

criteria and objectives not covered by the theorems presented in this article.

Based on the experimental results we may conclude that the tightened domination method significantly lowers the number of labels generated for each problem. Using better data structures which make it possible to quickly test for domination may further improve the running times.

Being able to handle non-additive objectives, the shortest path algorithms can more widely be applied to real-life problems. We believe that the tightened domination criteria given in this paper can be a tool box for others when defining such criteria for other real-life problems. Our method is a contribution in the process of optimizing more and more real-life problems and thereby using our resources optimally and minimizing waste both in the literal and metaphorical sense.

Acknowledgments

The authors wish to thank Patrick Lincoln, Julia Lawall, Amelia Regan and two anonymous referees for valuable comments.

Bibliography

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] L. Blander-Reinhardt. Multi-objective shortest path for cargo transportation. Master's thesis, University of Copenhagen, DIKU, Denmark, April 2005.

[3] J. Brumbaugh-Smith and D. Shier. An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43:216–224, 1989.

[4] R. Carraway, T. L. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research*, 44:95–104, 1990.

[5] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press/McGraw-Hill Book Company, 1997.

[6] G. Desaulniers, J. Desrosiers, I. Ioachim, M. Solomon, F. Soumis, and D. Villeneuve. *Fleet Mangaement and Logistics*. Kulwer Academic Publisher, 1998. chapter: Unified Framework for Deterministic Time Constrained Vehice Routing and Crew Scheduling Problems.

[7] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42:135–153, 2003.

[8] M. Ehrgott and X. Gandibleux. An annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum*, 22:425–460, 2000.

[9] C. Gueguen, P. Dejax, M. Dror, and M. Gendreau. An exact algorithm for the elementary shortest path problem with resource constraints. Technical report, Laboratoire Productique Logistique, Ecole Centrale Paris, 1998. revised in July 1999, also with D. Feillet as co-author.

[10] P. Hansen. Bicriteria path problems. *G. Fandel and T. Gal: Multi Criteria Decision Making Theory and Applications, Lecture Notes in Economics and Mathematical Systems*, 177:109–127, 1979.

[11] HUR. http://trafikinfo.hur.dk/priserogbilletter.

[12] K. Hutson and D. Shier. Extended dominance and a stochastic shortest path problem. *Computers & Operations Research*, 36:584–596, 2009.

[13] I. Ioachim, S. Gelinas, J. Desrosiers, and F. Soumis. A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31:193–204, 1998.

[14] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In M. M. S. G. Desaulniers, J. Desrosiers, editor, *Column Generation*, pages 33–65. Springer, USA, 2005.

[15] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k > 3$. *Informs Journal on Computing*, 18:391–406, 2006.

[16] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56:497–511, 2008.

[17] T. Lengauer and D. Theune. Efficient algorithms for path problems with gernal cost citeria. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 314 – 326, 1991.

[18] E. Martins. On a multicriteria shortest path problem. *European Journal of Operation Research*, 16:236–237, 1984.

[19] M. Müller-Hannemann and M. Schnee. Finding all attractive train connections by multi-criteria pareto search. In *Proceedings of the 4th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2004) Bergen, Norway, Lecture Notes in Computer Science*, 2004.

[20] M. Müller-Hannemann and K. Weihe. On the cardinality of the pareto set in bicriteria shortest path problems. *Annals of Operation Research*, 147:185–197, 2004.

[21] W. Powell and Z. Chen. A generalized threshold algorithm for the shortest path problem with time windows. In P. Pardalos and D. Du, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 303–318. American Mathematical Society, 1998.

[22] S. Skriver and K. Andersen. A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27:507–524, 2000.

[23] H. Taha. *Operations Research - An introduction.* Pearson Prentice Hall, eight edition edition, 2007.

[24] G. Tsaggouris and C. Zaroliagis. Non-additive shortest paths. In *Algorithms - ESA 2004. 12th Annual European Symposium*, volume 3221 of *Lecture Notes in Computer Science*, pages 822–834, 2004.

[25] G. Tsaggouris and C. Zaroliagis. Multiobjective optimization: improved fptas for shortest paths and non-linear objectives with applications. In *Algorithms and Computation. 17th International Symposium, ISAAC 2006*, volume 4288 of *Lecture Notes in Computer Science*, pages 389–398, 2006.

[26] D. Villeneuve and G. Desaulniers. The shortest path problem with forbidden paths. *European Journal of Operational Research*, 165:97–107, 2005.

Chapter 5

A Branch and Cut algorithm for the container shipping network design problem

Line Blander Reinhardt*[1] David Pisinger*[2]

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
lbre@man.dtu.dk, pisinger@man.dtu.dk

Abstract

The network design problem in liner shipping is of increasing importance in a strongly competitive market where potential cost reductions can influence market share and profits significantly. In this paper the network design and fleet assignment problems are combined into a mixed integer linear programming model minimizing the overall cost. To better reflect the real-life situation we take into account the cost of transhipment, a heterogeneous fleet, route dependent capacities, and butterfly routes. To the best of our knowledge it is the first time an *exact* solution method to the problem considers transhipment cost. The problem is solved with branch-and-cut using clover and transhipment inequalities. Computational results are reported for instances with up to 15 ports.

5.1 Introduction

Liner shipping routes are characterized by the cyclic routes repeatedly sailed during the scheduled horizon and the transhipment of cargo in hub ports. The process of designing the route network of a liner shipping company is essential for the competitiveness of the company and its ability to sustain and possibly improve the share of the global containerized freight market. The problem of determining the structure of the route network we call the liner shipping network design problem (LS-NDP). Designing efficient routes can reduce the overall cost and the CO_2 emission per container shipped.

To provide a competitive product a liner shipping company must at a minimal cost be able to satisfy the requests from customers for shipment of containers. Liner shipping companies usually have a forecast period over which the shipping demands are predicted based on historic data

[1]This research is partly supported by the Danish Maritime Fund
[2]This research was supported by the Danish Council for Strategic Research (ENERPLAN)

and recent development. The LS-NDP consists of designing vessel routes so that the forecasted requests are satisfied with a minimal cost for the company.

A vessel will repeatedly sail the assigned route throughout the entire planning horizon. This means that the routes are cyclic and the capacity of a link on a route depends on the number of times the link is sailed in the planning horizon.

The LS-NDP gained increasing attention about three decades ago when the container freight started to increase significantly. Recently the interest in the area has further increased due to the large focus on CO_2 emission generated by the vessels, and the dramatic change in demands created by the current financial crisis, which has resulted in a need to focus on lowering the costs.

The similarity between LS-NDP, network design and routing problems leads us to the assumption that methods that work well for other scheduling and network design problems will also work well for the LS-NDP. An example of such a method is the branch-and-cut method which has been successfully applied to the vehicle routing problem with time windows (VRPTW) problem, see Bard et al. [4] and Kallehauge et al. [12].

In this paper we present a mathematical formulation of the problem which includes transhipment, transhipment cost and allows a mix of simple and *butterfly* routes. An exact method using branch-and-cut has been developed for solving the presented model. The developed branch-and-cut method has been run on a set of test instances and compared to the CPLEX MIP solver. To our knowledge it is the first time an exact method has been applied to a problem which includes transhipment and the results show that small instances can be solved to optimality. The developed branch-and-cut method clearly outperforms CPLEX. The test results presented in the computational experiments, Section 7.7, document that the developed algorithm can be used for planning the routes of a smaller shipping company or a concrete region of the network of a bigger liner shipping company. The LS-NDP problems we solve to optimality are comparable in size to the test instances presented in recent literature on shipping network design using heuristic solution methods (see Agarwal and Ergun [1] and Alvarez [2]).

We will start with a literature review in Section 5.2. In Section 5.3, the problem is formulated as a graph theoretical problem and a mathematical model of the LS-NDP is presented. In Section 5.4 the branch-and-cut algorithm is described, and separation algorithms for the introduced transhipment cuts and connectivity cuts are presented. In Section 7.7, the tests and results are discussed. Finally, we make some concluding remarks and suggest areas for further research in Section 5.6.

5.2 Literature review

In this section we summarize the literature which has been used directly in our work. For a detailed literature review of cargo shipping optimization problems we refer the reader to the survey papers, Ronen [18], Ronen et al. [19], and Christiansen et al. [6]. The reader is also referred to Christiansen et al. [5] for a comprehensive introduction to the areas of optimization in maritime transportation.

In 1991 Rana and Vickson [17] presented a state-of-the-art model for container shipping on the North Atlantic trade routes. They worked with an outbound-inbound principle which, until recently, has been a standard principle in the liner shipping industry. The outbound-inbound principle means that the ports are listed in a predefined order and that a vessel goes through the list in one direction visiting selected ports and upon return goes through the list in the reverse direction until reaching the first port visited on the list. The liner shipping companies still have an inbound-outbound way of viewing some of their routes. However, there is no requirement that the routes must be scheduled this way. For shipping routes along a somewhat straight coastline such as the US West Coast investigated in [17] this is a natural setup. For inter continental routes or routes in enclosed seas such as the Baltic, Mediterranean and Black Sea the overall structure is usually not inbound-outbound. As a result, better routes may be found by relaxing the inbound-outbound restriction.

Rana and Vickson [17], Christiansen and Nygreen [7], Fagerholt [8], Agarwal and Ergun [1] and Alvarez [2] allow for several visits to a port. The allowance of several visits to a port is, in the

mentioned papers (with the exception of [1]) achieved by combining simple routes. In a simple route each port is visited at most once. Agarwal and Ergun [1] solve the problem by using a time-space graph where a port can be visited several times as long as the visit is not on the same weekday.

The shipping companies often wish to schedule the frequency of a departure at a port so that it corresponds to the demand at the port. Fagerholt [8] and Christiansen and Nygreen [7], and Agarwal and Ergun [1] have a weekly frequency requirement on the routes. Fagerholt [8] and Christiansen and Nygreen [7] formulate the weekly frequency by restricting the time of an route to be less than a week. This is applicable to small shipping routes such as regional routes. However, it is clear that when it comes to intercontinental shipping the routes are usually longer than a week. This is handled in Agarwal and Ergun [1] by covering the weekly departures with a sufficient number of vessels of the same type.

The use and influence of transhipment on the liner shipping network design is described by Notteboom and Rodrigue [16]. However only a few decades ago the use of transhipment was much less common. Therefore older articles such as Rana and Vickson [17] do not include transhipment in their route planning. The model solved in [17] was extended in the recent work by Shintani et al. [20], where the restrictive visiting order of Rana and Vickson [17] is relaxed as to represent a more realistic set of routes. Moreover, the repositioning of empty containers is included by Shintani et al. [20]. To solve the problem presented in [20] a genetic algorithm is used, however, transhipment is not considered. Christiansen and Nygreen [7] use column generation to solve the routing problem for ammonia shipping in Norway. In the problem solved in [7] only ammonia is shipped and therefore transhipment is not considered. Fagerholt [8] apply column generation for solving the liner shipping problem along the Norwegian coast. Others, such as Gelareh and Meng [10] exclusively deal with the fleet deployment on a predefined set of routes. In the recent paper by Agarwal and Ergun [1], the authors solve larger problems by using a heuristic based on Benders' Decomposition and compare it to a similar solution method which uses column generation. Recently an article on liner shipping network design optimization has been publish by Alvarez [2] using tabu search and column generation. The model has transhipment costs as part of the overall cost evaluation. Rana and Vickson [17], Christiansen and Nygreen [7] and Fagerholt [8] do not consider transhipment. Even though Notteboom and Rodrigue in [16] emphasize the importance of transhipment in the shipping networks, Agarwal and Ergun in [1] are the first to include transhipment in the liner shipping network design problem. However, they do not include transhipment cost and Alvarez in [2] from 2009 is to our knowledge the first to consider the cost of transhipment when designing the shipping network. In models where each port is represented by one vertex at the points where two cycles are connected a transhipment from an early visit of a vessel to a later visit of the same vessel can occur. As will be discussed in Section 5.3.2 this results in complications in the calculation of transhipment costs. To our knowledge the exact cost of transhipment has not been calculated at the cycle connection points earlier. Note that Argawal and Ergun [1] do not use a single vertex for representing a port and that they do not include transhipment cost. Clearly, increasing the number of vertices and thereby the number of edges in the graph will significantly increase the complexity of the problem even though it gives more flexibility in the route structure.

The models by Agarwal and Ergun [1] and Alvarez [2] are so far the most comprehensive representations of the problem faced by liner shipping companies. Alvarez [2] include many relevant parameters in the objective while Agarwal and Ergun [1] only include cost.

Even though shipping companies often have several vessels of the same type it is not always an optimal solution to force the routes to be sailed by the same vessel type and in real-life routes there are some smaller ports which, due to low demand, only require a bimonthly departure and some busy ports might require a biweekly departure.

To the best of our knowledge no results for the LS-NDP, using branch-and-cut, have been presented in the literature. As mentioned in Section 5.1 good results have been achieved by [4] and [12] when applying branch-and-cut to the VRPTW. Since the VRPTW is somewhat similar to the LS-NDP with its heterogeneous fleet and cyclic routes it is natural to assume that branch-and-cut will also result in good solutions for the LS-NDP.

5.3 Problem formulation

Let G be a directed graph and let $G(\mathbf{N}, \mathbf{A}, \mathbf{V}, \mathbf{M}, t_{max})$ represent the network with vertex set $\mathbf{N}$, arc set $\mathbf{A}$, a set of vessels $\mathbf{V}$, a set of demands $\mathbf{M}$ and a forecast period with length t_{max}. Each vertex $n \in \mathbf{N}$ represents a port. Each arc $(i, j) \in \mathbf{A}$ is a direct connection between two ports for a given vessel $v \in \mathbf{V}$. Each demand $m \in \mathbf{M}, m = (i^m, j^m, d^m, t^m)$ is the amount $d^m \in \mathbb{Z}$ of type t^m to be shipped from an origin port i^m to a destination port j^m. Even though the container type is ignored in the tests it can easily be included and is relevant as the cost of shipping and transhipping a reefer container can be very different from the cost of a normal container.

Each vertex j has a cost of transhipping demand m, depending on the type of demand, c_j^m and a service time t_j. Each arc a has a cost c_{ij}^m of carrying demand m on a direct connection from port i to port j. Each arc (i, j) also has an associated time t_{ij}^v reflecting the prefixed time it takes for vessel v to sail a direct connection from port i to port j. Each vessel $v \in \mathbf{V}$ has a capacity C^v. The liner shipping network design problem is to find a connected route for each vessel $v \in \mathbf{V}$ where the customer demands are satisfied and the overall cost is minimized. Since a vessel assigned a route sails continuously during the whole planning horizon, the cost to be minimized is a linear function of the cost of using a selected vessel, the cost of transporting a demand on the arcs and the cost of transhipping at ports. It can be argued that the cost of transporting a demand is negligible. However by introducing a small cost corresponding to time, one can be assured that unnecessary extra time or travel is avoided for the demand. In the cost of transporting a demand we only include the time the demand spend on the vessel and not the time the demand uses at a port during transhipment. In the model the objective is to minimize the overall cost so that the required demand can be shipped from their origin to their destination within the time interval of length t_{max}.

Figure 5.1 shows an example of a network containing two liner shipping routes. Both routes visit a port twice. Routes visiting a single port twice are denoted *Butterfly* routes. In Figure 5.1 he edges of each route are numbered in sailing order starting at the port visited twice. In the figure, transhipment can occur at the ports A,B and C. At ports A and B transhipment can occur between the two routes moreover at port A transhipment can occur between two visits of route 2 and at port C transhipment can occur between two visits of route 1.

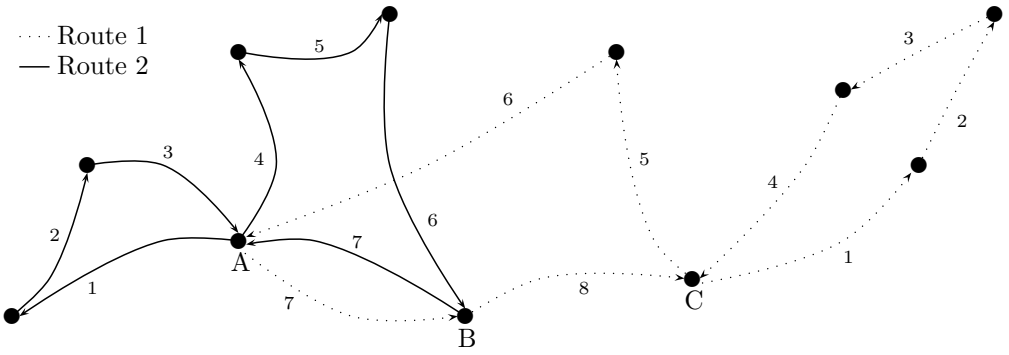


Figure 5.1: An example of two routes in a liner shipping network. Each route can be constructed by following the arcs in increasing order starting with arc number 1. Transhipment can take place at port A, B and C.

5.3.1 Mathematical Model

There is no standard mathematical formulation of the liner shipping problem since each liner shipping company has specific constraints based on strategic decisions. As a result of this, several formulations and models have been presented in the literature.

In this section we first present a comprehensive mathematical model for the liner shipping problem which includes transhipment, transhipment cost, simple routes, butterfly routes and a heterogeneous vessel fleet.

5.3.2 The Network Design Problem

In the model presented by Agarwal and Ergun in [1] a time-space graph structure is used, where the time is the day of week and weekly departures by vessels of the same type is enforced on the same weekday. However, in the here presented version of the liner shipping problem a port is allowed to be visited less than once a week and different vessel types are permitted to sail the same route. Allowing for other than weekly departure and different vessel types on a route may result in lower cost. Moreover, in the model presented here, we have chosen to have a cost for using a vessels, as opposed to Agarwal and Ergun in [1] who use a cost for sailing a route. Shipping companies usually wish to cover the demand with the least number of vessels. The model is presented first and in the following subsections selected areas of the model are described.

We have the following variables:

x_{ij}^{mv} the amount of demand m shipped on arc (i,j) by vessel v,

u_{ij}^{v} a binary variable which is 1 if arc (i,j) is the first or the last arc on a route of vessel v with two loops, 0 otherwise,

e_{ij}^{v} a positive integer variable enumerating the order of the arcs on the route for v, $e_{ij}^{v} \in \{0,\ldots,|\mathbf{N}|\}$,

y_{ij}^{v} a binary variable which is 1 if arc (i,j) is in the route of vessel v, 0 otherwise,

f_{j}^{mv} the amount of demand m from vessel v transhipped at port j,

s_{i}^{v} a binary variable which is 1 if i is the port which may connect two loops for vessel v, denoted centerpoint,

f_{jih}^{mv} the amount of demand m from vessel v entering i from j and not leaving on the arc from i to h,

τ_{v} the route travel time of vessel v,

h^{v} a binary variable which is 1 if vessel v is sailing and 0 otherwise,

We use the following parameters:

C^{v} the capacity of vessel v,

t_{ij}^{v} the time it takes for vessel v to sail arc (i,j),

t_{max} the duration of the forecast period,

t_{j} the time at quay at port j.

The demands are defined as:

$$b_i^m = \begin{cases} d^m & \text{if } i \text{ is origin of demand } m \\ -d^m & \text{if } i \text{ is destination of demand } m \\ 0 & \text{otherwise} \end{cases} \qquad m \in \mathbf{M}, i \in \mathbf{N}$$

The four "big-M" coefficients M_1,M_2,M_3 and M_4 are sufficiently large constants. We operate with three different costs: c^v the cost of vessel v sailing, c_{ij}^m the unit cost of shipping demand m on connection (i,j), and c_i^m the unit cost of transhipping demand m at port i. This leads to the model:

$$\textbf{Min:} \sum_{m\in\mathbf{M}}\sum_{(i,j)\in\mathbf{A}} c_{ij}^{m}\sum_{v\in\mathbf{V}} x_{ij}^{mv} + \sum_{m\in\mathbf{M}}\sum_{j\in\mathbf{N}} c_{j}^{m}\sum_{v\in\mathbf{V}} f_{j}^{mv} + \sum_{v\in\mathbf{V}} c^{v}h^{v} \tag{5.1}$$

s.t.

$$\text{(Flow)} \quad \sum_{v\in\mathbf{V}}\sum_{j:(i,j)\in\mathbf{A}} x_{ij}^{mv} - \sum_{v\in\mathbf{V}}\sum_{j:(j,i)\in\mathbf{A}} x_{ji}^{mv} = b_{i}^{m} \qquad i\in\mathbf{N}, m\in\mathbf{M} \tag{5.2}$$

$$\text{(Trans 0)} \quad f_{i}^{mv} \geq \sum_{j:(j,i)\in\mathbf{A}} x_{ji}^{mv} - \sum_{j:(i,j)\in\mathbf{A}} x_{ij}^{mv} \qquad m\in\mathbf{M}, i\in\mathbf{N}, v\in\mathbf{V} \tag{5.3}$$

$$\text{(Trans 1)} \quad f_{i}^{mv} \geq \sum_{j,h\in\mathbf{N},v\in\mathbf{V}} f_{jih}^{mv} - M_1(1-s_{i}^{v}) \qquad m\in\mathbf{M}, i\in\mathbf{N}, v\in\mathbf{V} \tag{5.4}$$

$$\text{(Trans 2)} \quad f_{jih}^{mv} \geq x_{ji}^{mv} - x_{ih}^{mv} - M_2(2-y_{ji}^{v}-y_{ih}^{v}+u_{ji}^{v}+u_{ih}^{v}) \qquad m\in\mathbf{M}, j,i,h\in\mathbf{N}, v\in\mathbf{V} \tag{5.5}$$

$$\text{(Trans 3)} \quad f_{jih}^{mv} \geq x_{ji}^{mv} - x_{ih}^{mv} - M_3(4-u_{ji}^{v}-u_{ih}^{v}-y_{ji}^{v}-y_{ih}^{v}) \qquad m\in\mathbf{M}, j,i,h\in\mathbf{N}, v\in\mathbf{V} \tag{5.6}$$

$$\text{(Capacity)} \quad \frac{t_{max}}{\tau_{v}} C^{v} y_{ij}^{v} \geq \sum_{m\in\mathbf{M}} x_{ij}^{mv} \qquad (i,j)\in\mathbf{A}, v\in\mathbf{V} \tag{5.7}$$

$$\text{(Center)} \quad \sum_{i\in\mathbf{N}} s_{i}^{v} = 1 \qquad v\in\mathbf{V} \tag{5.8}$$

$$\text{(First arc)} \quad \sum_{(i,j)\in\mathbf{A}} u_{ij}^{v} = 2 \qquad v\in\mathbf{V} \tag{5.9}$$

$$\text{(Out arc)} \quad s_{i}^{v} - \sum_{j:(i,j)\in\mathbf{A}} u_{ij}^{v} \leq 0 \qquad i\in\mathbf{N}, v\in\mathbf{V} \tag{5.10}$$

$$\text{(In arc)} \quad s_{i}^{v} - \sum_{j:(j,i)\in\mathbf{A}} u_{ji}^{v} \leq 0 \qquad i\in\mathbf{N}, v\in\mathbf{V} \tag{5.11}$$

$$\text{(Cyclic)} \quad \sum_{j:(i,j)\in\mathbf{A}} y_{ij}^{v} - \sum_{j:(j,i)\in\mathbf{A}} y_{ji}^{v} = 0 \qquad i\in\mathbf{N}, v\in\mathbf{V} \tag{5.12}$$

$$\text{(Connect 0)} \quad \sum_{j:(i,j)\in\mathbf{A}} y_{ij}^{v} - s_{i}^{v} \leq 1 \qquad i\in\mathbf{N}, v\in\mathbf{V} \tag{5.13}$$

$$\text{(Connect 1)} \quad e_{ji}^{v} - e_{ih}^{v} + M_4(y_{ih}^{v}+y_{ji}^{v}-2-u_{ji}^{v}-u_{ih}^{v}) \leq -1 \qquad i,j,h\in\mathbf{N}, v\in\mathbf{V} \tag{5.14}$$

$$\text{(Ships)} \quad y_{ij}^{v} - h^{v} \leq 0 \qquad (i,j)\in\mathbf{A}, v\in\mathbf{V} \tag{5.15}$$

$$\text{(Time 0)} \quad \tau_{v} \leq t_{max} \qquad v\in\mathbf{V} \tag{5.16}$$

$$\text{(Time 1)} \quad \tau_{v} = \sum_{i,j:(i,j)\in\mathbf{A}} y_{ij}^{v}\left(t_{ij}^{v}+t_{j}\right) \qquad v\in\mathbf{V} \tag{5.17}$$

$$u_{ij}^{v}, y_{ij}^{v} \in \{0,1\} \qquad (i,j)\in\mathbf{A}, v\in\mathbf{V} \tag{5.18}$$

$$f_{jih}^{mv} \geq 0 \qquad m\in\mathbf{M}, j,i,h\in\mathbf{N}, v\in\mathbf{V} \tag{5.19}$$

$$f_{j}^{mv} \geq 0 \qquad m\in\mathbf{M}, j\in\mathbf{N}, v\in\mathbf{V} \tag{5.20}$$

$$e_{ij}^{v} \in \mathbb{Z}^{+} \qquad i,j\in\mathbf{N}, v\in\mathbf{V} \tag{5.21}$$

$$x_{ij}^{mv} \geq 0 \qquad (i,j)\in\mathbf{A}, m\in\mathbf{M}, v\in\mathbf{V} \tag{5.22}$$

$$s_{i}^{v} \in \{0,1\} \qquad i\in\mathbf{N}, v\in\mathbf{V} \tag{5.23}$$

$$h^{v} \in \{0,1\} \qquad v\in\mathbf{V} \tag{5.24}$$

$$\tau_{v} \geq 0 \qquad v\in\mathbf{V} \tag{5.25}$$

The objective (5.1) minimizes the sum of the cost of transporting the demand, the cost of transhipping demand and the cost of using the vessels. Constraints (5.2) ensure flow conservation and that all demand $m \in M$ is satisfied. Constraints (5.3) ensure that f_{i}^{mv} is larger than the difference between the incoming demand m and outgoing demand m on a vessel v. Since the objective is to minimize the cost and $c_{i}^{m} f_{i}^{mv}$ is positive then f_{i}^{mv} will be equal to the amount

transhipped. Constraints (5.4), (5.5) and (5.6) together with the constraints (5.9), (5.10),(5.11) and (5.14) for u_{ij}^v ensure that f_i^{mv} at the vertex i connecting two loops sailed by the same vessel also includes the amount left at the port to be picked up later by the same vessel.

The capacity constraints (5.7) ensure that the amount shipped on vessel v on arc (i, j) is less than the capacity of the vessel v multiplied by the number of trips which can be completed in the schedule period t_{max}. Note that the value of τ_v is determined in constraints (5.17) where the right hand side is the time of the route sailed by vessel v. The constraints (5.8) ensure that for each route exactly one vertex is selected as centerpoint. Note, that a centerpoint is the port which may be visited multiple times, although it is not required to be visited more than once. The constraints (5.12) ensure that for every port, every vessel which enters the port also leaves the port. Constraints (5.13) ensure that a vessel v does not visit its selected start port more than twice.

Constraints (5.14) ensure that all parts of a route sailed by vessel v is connected to the start port s_i^v of vessel v. if $y_{ji}^v = 1$, $y_{ih}^v = 1$, $u_{ji}^v = 0$ and $u_{ih}^v = 0$, then the variable e_{ih}^v is one greater than variable e_{ji}^v. Otherwise, e_{ih}^v can be any positive integer satisfying constraint (5.14). Constraints (5.15) ensure that there will be a cost for using vessel v in the objective. Constraints (5.16) ensure that no route is longer than the schedule period.

In the following sections we will discuss how the requirements special to the LS-NDP can be formulated in a linear model.

Transhipment cost in the Liner Shipping Network Design

Agarwal and Ergun [1] argued that transhipment is the core of liner shipping. We would like to add that transhipment of goods is frequently occurring in liner shipping and the associated cost should not be ignored when designing the network. Transhipment are allowed in the model presented in [1], however the expenses of transhipment were not included in the cost calculation before the work by Alvarez in [2]. To calculate the transhipment cost when satisfying demands in a specific network design one must know the amount of containers, which is transhipped. We define a variable f_i^{vm} which is the amount of containers in demand m transhipped at port i from vessel v. In the objective function (5.1) the cost c_j^m of transhipping one unit at port i is included. To find the value of f_i^{vm} we have the constraints (5.3). When the routes are simple the amount transshipped can be calculated by constraint (5.3) alone. However when *butterfly* routes exists there can be cargo transhipped at the centerpoint which is not calculated by the constraint (5.3). This cargo is the containers transhipped between two visits of the same route to the port. Therefore, to ensure that a cost is charged for the containers transhipped between two visits of the same route at a port, it is important to be able to distinguish between the two visits to the centerpoint. This can be achieved by enumerating the edges on the route and marking the first and last edge on the entire route. The integer variables e_{ij}^v enumerates the edges on the route and the binary variables u_{ij}^v marks the first and last edge on a *butterfly* route. The following constraints ensure that the first and last edge on a *butterfly* route are found:

$$\text{(First arc)} \qquad \sum_{(i,j)\in\mathbf{A}} u_{ij}^v = 2 \qquad v \in \mathbf{V} \qquad (5.26)$$

$$\text{(Out arc)} \qquad s_i^v - \sum_{(ij)\in\mathbf{A}} u_{ij}^v \leq 0 \qquad i \in \mathbf{N}, v \in \mathbf{V} \qquad (5.27)$$

$$\text{(In arc)} \qquad s_i^v - \sum_{(ji)\in\mathbf{A}} u_{ji}^v \leq 0 \qquad i \in \mathbf{N}, v \in \mathbf{V} \qquad (5.28)$$

$$\text{(Connect 1)} \qquad e_{ji}^v - e_{ih}^v + M_4(y_{ih}^v + y_{ji}^v - 2 - u_{ji}^v - u_{ih}^v) \leq -1 \qquad i, j, h \in \mathbf{N}, v \in \mathbf{V} \qquad (5.29)$$

Where s_i^v is the port selected as centerpoint for the route. The constraints (5.29) ensure that if the route is a *butterfly* route then the last edge (j, i) on the route has $u_{ji}^v = 1$ and the first edge (i, h) on the route has $u_{ih}^v = 1$.

To find the demand transhipped from one visit to another visit of the same vessel we introduce the variable f^{mv}_{jih} indicating the transhipment in i when arriving form port j and departing to port h. Then on a *butterfly* route the two visits to a centerpoint i are the one where $u^v_{ji} = u^v_{ih} = 1$ and $y^v_{ji} = y^v_{ih} = 1$ and the one where $u^v_{ki} = u^v_{il} = 0$ and $y^v_{ki} = y^v_{il} = 1$ Clearly if $u^v_{ji} = u^v_{ih} = 1$ and $y^v_{ji} = y^v_{ih} = 1$ then the transhipment at one visit to the port i is $x^{mv}_{ji} - x^{mv}_{ih}$, which can be formulated as:

$$\text{(Trans 3)} \quad f^{mv}_{jih} \geq x^{mv}_{ji} - x^{mv}_{ih} - M_3(4 - u^v_{ji} - u^v_{ih} - y^v_{ji} - y^v_{ih}) \quad m \in \mathbf{M}, j, i, h \in \mathbf{N}, v \in \mathbf{V} \tag{5.30}$$

If $u^v_{ki} = u^v_{il} = 0$ and $y^v_{ki} = y^v_{il} = 1$ then the transhipment at one visit to the port i is $x^{mv}_{ki} - x^{mv}_{il}$, which can be formulated as:

$$\text{(Trans 2)} \quad f^{mv}_{kil} \geq x^{mv}_{ki} - x^{mv}_{il} - M_2(2 - y^v_{ki} - y^v_{il} + u^v_{ki} + u^v_{il}) \quad m \in \mathbf{M}, k, i, l \in \mathbf{N}, v \in \mathbf{V} \tag{5.31}$$

This must be included in the value of the f^{mv}_i used in the objective. However the f^{mv}_i only need to be adjusted for the centerpoint of the route. The port i is a centerpoint if $s^v_i = 1$. At the centerpoint the transhipment amount is the sum of the transhipment on the two visits. This can be formulated as:

$$\text{(Trans 1)} \qquad f^{mv}_i \geq \sum_{j,h \in \mathbf{N}, v \in \mathbf{V}} f^{mv}_{jih} - M_1(1 - s^v_i) \qquad m \in \mathbf{M}, i \in \mathbf{N}, v \in \mathbf{V} \tag{5.32}$$

Constraints (5.30) calculates the amount unloaded from the vessel at the visit to port i from the end edge to the start edge of the route. Constraints (5.31) calculates the amount unloaded from the vessel at the other visit to port i. Constraints (5.32) ensure that this is included in the transhipment on route v at a centerpoint i. The number of constraints in (5.31) and (5.30) is $O(|N^3||M||V|)$ which is a significantly large number. The additional binary and integer variables u^v_{ij} and e^v_{ij} may increase the size of the branch-and-bound tree.

The cyclic structure of liner shipping routes

In the liner shipping network design problem a vessel must leave each port it enters. This is called flow conservation and is modeled by constraints (5.12).

Clearly it is important to ensure that a route is connected so that it can be sailed by a single vessel and avoid having several disconnected subtours, for example two separate cycles, representing a route. When modeling the constraint that the route must be connected it is often assumed that the route is simple. Although the routes are simple in [1], the time-space graph used by Agarwal and Ergun in [1] allows for multiple visits to a port as long as the visits do not happen on the same day of the week.

In the model presented here we let each route have a port which may be visited at most twice, in order to model the real life situation where a port is used as a hub. Notteboom notes in [15] that this form of design is used by Maersk Line. This can also be confirmed by studying the mapped services of Maersk, available online (see [14]). Figure 5.2 shows the distribution of the route types in [14], where we have looked apart from routes which are presented as one-way strings. There are 58 connected routes, of which 42 routes were simple or butterfly routes and the remaining 16 routes contained multiple ports with more than one visits. This shows that 72% of the routes studied are simple or butterfly routes.

To model *butterfly* routes the binary variable s^v_i is introduced indicating which port on the route may be used as Hub. To have a polynomial number of constraints ensuring that the routes are connected we have used the approach proposed by Tucker et al. [13] for enumerating the vertices on a simple path. In constraints (5.14) the arcs on the route are enumerated instead of the ports using the u^v_{ij} variables to mark the start and end arc of the route. The constraints presented by Tucker et al. [13] are used in vehicle routing problems and variances with simple routes. However in the presented model for the LS-NDP a single port on the route may be visited twice. Allowing the possibility of two visits to the hub port it could be formulated as:

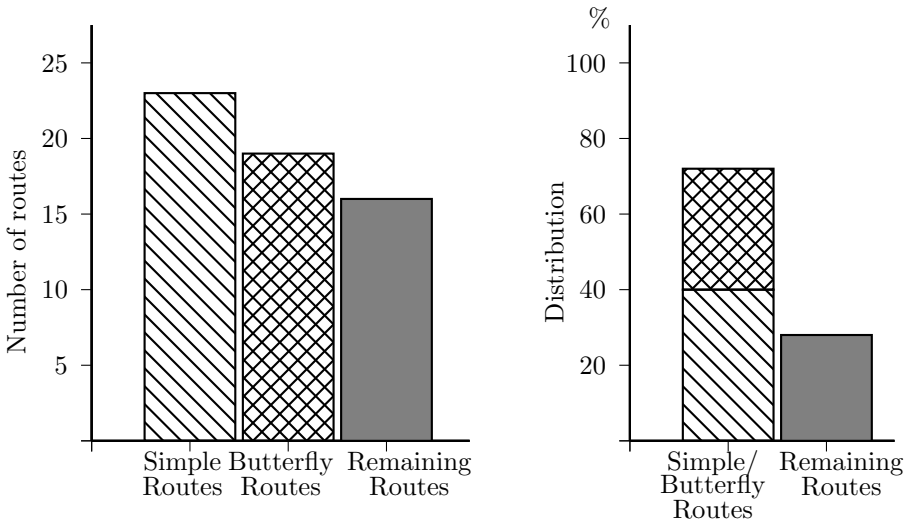


Figure 5.2: Histograms showing the distribution of the route types.

$$\text{(Center)} \qquad \sum_{i \in \mathbf{N}} s_i^v = 1 \qquad v \in \mathbf{V} \tag{5.33}$$

$$\text{(Butterfly)} \qquad z_i^v - z_j^v + M_4(y_{ij}^v - s_j^v) \leq M_4 - 1 \qquad i, j \in \mathbf{N}, v \in \mathbf{V} \tag{5.34}$$

Where z_j^v is a positive integer. The variable indicates the order in which the ports are visited on each loop seen in isolation, but does not indicate which loop is visited first. However, since the cost of transhipment is included as described in previous Section 5.3.2 it is needed to know the start and end edges of the route at the hub port. Therefore the constraint is formulated as:

$$\text{(Connect 1)} \qquad e_{ji}^v - e_{ih}^v + M_4(y_{ih}^v + y_{ji}^v - 2 - u_{ji}^v - u_{ih}^v) \leq -1 \qquad i, j, h \in \mathbf{N}, v \in \mathbf{V} \tag{5.35}$$

This means that constraints concerning s_i^v and u_{ij}^v must be included. Therefore to model connected *butterfly* routes while allowing for calculating the transhipment cost the constraints (5.9), (5.10), (5.11), (5.12), (5.13), (5.33) and (5.35) are needed. Note that there is an overlap with the constraints needed for calculating the exact transhipment cost.

The number of times a route can be completed in a schedule period

The number of times a link is sailed during the time period affects the capacity on the given link.

In our model a route can at most contain a link once. However, every link on a given route is sailed the number of times the route can be completed by the assigned vessel in the schedule period. Since a link can be sailed on several different routes, the number of times a link is sailed also depends on the number of routes the link appears in.

For example a vessel with the capacity to carry 1000 containers and sailing a route which takes 30 days can in a 30 day forecast period only ship 1000 containers on each leg of the route. However if the same vessel sailed a route which only takes 5 days it could on each leg of the route, ship 6000 containers during the same period. Therefore we include the route length in the capacity constraint in the LS-NDP model. The consideration of route length in liner shipping network design was first introduced by Agarwal and Ergun [1], where vessels of the same type are assigned to a route so that there is always a weekly departure. As mentioned earlier weekly departures are not a strict requirement for all shipping companies. In real-life shipping, ports with smaller demands are visited bi-monthly. Moreover it may happen that a shipping company does not have the right number of ships of a specific type to cover a weekly departure on a route. It is also likely that a better solution has different vessel types assigned to a route.

To include the time of the route in the calculation of the capacity, we multiply the capacity of a vessel with the number of times the route can be completed during the forecast period. This requirement is formulated by the constraints (5.7) where the partial route is included in the capacity calculation as a partial vessel capacity. These constraints are not linear and thus to solve this problem using an integer programming solver it is necessary to linearize the constraints.

We here linearize the equation expressed by constraints (5.7). This is done by introducing the following variables:

q^v	t_{max}/τ_v, the schedule period divided by the route time.
r^v_{gh}	Some real number greater than the travel time of vessel v on arc (g,h) plus service time at port h multiplied by the times the route can be completed
M	Upper bound for the maximum capacity times maximum route time on any arc.

If $y^v_{ij} = 0$ the flow x^{mv}_{ij} must be equal to 0. Thus we introduce the constraints:

$$\text{(Capacity 2)} \quad \sum_{m \in \mathbf{M}} x^{mv}_{ij} \leq y^v_{ij} M \qquad (i,j) \in \mathbf{A}, v \in \mathbf{V} \tag{5.36}$$

These constraints ensure that nothing can be transported on an arc not included in a route. Now we look at the remaining case where the arc (i,j) is traversed. Since $y^v_{ij} = 1$ and $t^v_{ij} + t_j > 0$ then we know that $\tau_v > 0$.

We introduce the variable q^v so that :

$$t_{max}/\tau_v \geq q^v \quad v \in \mathbf{V} \tag{5.37}$$

Where $q^v \in \mathbb{R}^+_0$. Note that the constraint (5.37) is not linear. Since $\tau_v > 0$ for all $v \in \mathbf{V}$, we can express constraint (5.37) as:

$$t_{max} \geq q^v \tau_v \quad v \in \mathbf{V} \tag{5.38}$$

However, constraints (5.38) are still not linear. An entry in the sum over $(g,h) \in \mathbf{A}$ on the righthand side is $q^v \left(t^v_{gh} + t_h\right)$ when y_{gh} is one, and zero when y_{gh} is zero. Therefore to linearize this by the "Big M" method from [21] we write the following constraints:

$$\text{(Cap 3)} \qquad r^v_{gh} + M(1 - y^v_{gh}) - q^v(t_{gh} + t_h) \geq 0 \qquad (g,h) \in \mathbf{A}, v \in \mathbf{V} \tag{5.39}$$

$$\text{(route time)} \qquad t_{max} - \sum_{(g,h) \in \mathbf{A}} r^v_{gh} \geq 0 \qquad v \in \mathbf{V} \tag{5.40}$$

$$q^v \geq 0 \qquad v \in \mathbf{V} \tag{5.41}$$

$$r^v_{gh} \geq 0 \qquad (g,h) \in \mathbf{A}, v \in \mathbf{V} \tag{5.42}$$

Constraints (5.39) ensure that when y^v_{gh} is one then $r^v_{gh} \geq q^v(t_{gh} + t_h)$. When y^v_{gh} is zero then: $r^v_{gh} \geq q^v(t_{gh} + t_h) - M$. Note that M must be chosen so that $q^v(t_{gh} + t_h) - M \leq 0$, and that constraints (5.40) can replace constraints (5.16) in the model.

Now we can let q^v replace t_{max}/τ_v in the constraint formulation (5.7) and thereby we get inequality:

$$q^v C^v y^v_{ij} \geq \sum_{m \in \mathbf{M}} x^m_{ij} \quad (i,j) \in \mathbf{A}, v \in \mathbf{V} \tag{5.43}$$

which we again must linearize. Here we note that the left hand side is equal to $q^v C^v$ when $y^v_{ij} = 1$ and zero otherwise. Hence

$$\text{(Capacity 1)} \qquad q^v C^v + M(1 - y^v_{ij}) \geq \sum_{m \in \mathbf{M}} x^{mv}_{ij} \qquad (i,j) \in \mathbf{A}, v \in \mathbf{V} \tag{5.44}$$

The constraints of type (5.44) ensure that the flow on all by v sailed arcs is less than $q^v C^v$. For all arcs not sailed by v constraints (5.44) does not add any restrictions given that M is chosen big enough. Recall that constraints (5.36) ensure that there is not assigned flow to arcs which are not sailed. We include constraints (5.36), (5.39), (5.40), (5.44) and variable definitions (5.41) and (5.42) to replace the non-linear capacity constraints (5.7) and constraints (5.16) and (5.17).

These constraints are included in the integer linear programming (ILP) model used in the test for the branch-and-bound and branch-and-cut method.

The Compact model for Liner Shipping

The linear model for the liner shipping problem, which here is named the *compact model* is the model presented in the beginning of this section where constraints (5.7), (5.16) and (5.17) are replaced by the constraints (5.36), (5.39), (5.40), (5.44) and variable definitions (5.41) and (5.42). As the name indicates the *compact model* has a polynomial number of constraints. Since this model is linear it can be solved directly by an ILP solver. The problem is NP hard as it includes the model [1] as a special case, and the 'big-M' constraints (5.4),(5.5),(5.6),(5.14),(5.39) and (5.44) together with the large number of variables make the problem hard to solve for ILP solvers.

5.4 The Solution Method

The *compact model* can be solved using branch-and-bound but the 'big-M' constraints may result in large integrality gaps and poor bounds resulting in large search trees. Moreover the many variables make the problem combinatorically hard. As mentioned earlier the branch-and-cut method has successfully been applied to vehicle routing problems (Ascheuer et al. [3]) and other transportation network design problem, (Gendreau et al. [11]). Therefore it is interesting to investigate the possibilities for using the branch-and-cut method on the LS-NDP and compare it with a branch-and-bound method.

5.4.1 Branch-and-cut

The branch-and-cut method generally gives good results on problems with complicating constraints such as non linear constraints or problems with an exponential number of constraints. As in Ascheuer et al. [3] and Gendreau et al. [11] we gradually add the transhipment and connectivity constraints to the formulation when they are violated.

Transhipment cuts

Calculating the amount unloaded from a vessel at a port to be loaded onto the same vessel at a later visit to the port is quite cumbersome. For calculating the transhipment to be picked up at a port by the same vessel we use the constraints (5.4),(5.5) and (5.6). Note that constraints (5.5) and (5.6) each represents $|N|^3|M||V|$ constraints. We wish to remove the constraints (5.4),(5.5) and (5.6) and introduce them as cuts when they are violated. Note that these constraint can only be violated in the model when the route is a *butterfly* route as transhipment can only be picked up at a port by the same vessel at the point the two loops meet on a *butterfly* route. The point where the two loops meet are the centerpoint of the route and it is indicated by $s_i^v = 1$.

We have constructed a cut so that if all arcs in a set of arcs T are sailed by a vessel v then if it is a butterfly route with the centerpoint $s_i^v = 1$, we have two arcs (j, i) and (i, h) in T which are not on the same loop that can be selected as the start and end arc of the route. This is formulated as:

$$u_{ji}^v + u_{ih}^v + 2(|T| - y^v(T)) \geq 2, \tag{5.45}$$

where the arcs (j, i) and (i, h) are the first and last arc on the route. The function $y^v(T)$ returns the arcs in T sailed by vessel v and $|T|$ is the number of arcs in the set T. For calculating the

transhipment between two visits by the same vessel on this route we add the following constraints as cuts:

$$\text{(Tranship 1)} \qquad f_i^{mv} \geq \sum_{j,h \in \mathbf{N}, v \in \mathbf{V}} f_{jih}^{mv} - M(1 - s_i^v) \tag{5.46}$$

$$\text{(Tranship 2)} \qquad f_{hij}^{mv} \geq x_{ji}^{mv} - x_{ih}^{mv} - M(2 - y_{ji}^v - y_{ih}^v + u_{ji}^v + u_{ih}^v) \tag{5.47}$$

$$\text{(Tranship 3)} \qquad f_{hij}^{mv} \geq x_{ji}^{mv} - x_{ih}^{mv} - M(4 - u_{ji}^v - u_{ih}^v - y_{ji}^v - y_{ih}^v) \tag{5.48}$$

Note that for each of the constraints (5.45) added one of each constraint (5.46), (5.47) and (5.48) is added.

Connectivity cuts

In the network design cases where branch-and-cut has been applied it is assumed that routes are simple. For simple routes in the generalized traveling salesman problem the connectivity constraints have been formulated by Fischetti et al. [9] as:

$$\sum_{i,j \in S} y_{ij}^v \leq \sum_{h \in \mathbf{N}, g \in S \setminus \{k\}} y_{hg}^v - \sum_{e \in \mathbf{N}} y_{el}^v + 1 \qquad v \in \mathbf{V}\ \emptyset \subset S \subset \mathbf{N}, k \in S, l \in \mathbf{N} \setminus S \tag{5.49}$$

As mentioned before the real routes of the shipping companies are often not simple and we have introduced the concept of *butterfly* routes in Section 5.3.2. This extension introduces new and weaker connectivity constraints. Since the first and last edge are selected by the transhipment cuts (see Section 5.4.1) the edge order can be ignored here.

The connectivity constraints from [9] have been modified to allow *butterfly* routes. We will present a connectivity cut which will allow for $\psi + 1$ subtours which all have exactly one point in common. Connected routes with $\psi + 1$ subtours which all have exactly one point in common are denoted *pseudo-simple* routes. Because the subtours must go through exactly one common point, we call this type of cut a *clover-cut*. The clover-cut removes any route which is not connected but it keeps routes which are *pseudo-simple*.

Lemma 1. *For any cyclic disconnected* clover *path there exists a set S, and two vertices k and l for which the following clover-cut inequality is violated. Moreover no S,k and l violating a connected* clover *path exists. This can be expressed as:*

$$\sum_{i,j \in S} y_{ij}^v \leq \sum_{h \in \mathbf{N}, g \in S \setminus \{k\}} y_{hg}^v + \psi s_k^v - \sum_{e \in \mathbf{N}} y_{el}^v + \psi s_l^v + 1 \qquad v \in \mathbf{V}, \emptyset \subset S \subset \mathbf{N}, k \in S, l \in \mathbf{N} \setminus S \tag{5.50}$$

Proof. Let v_1 be the route. First we prove that we cannot find a S, k and l for which the clover-cut inequality does not hold for a connected *pseudo-simple* route.

Case 1 on S: Assume that there is no part of the route v_1 outside of S, where v_1 is a connected *pseudo-simple* route. Then $\sum_{e \in \mathbf{N}} y_{el}^{v_1} = 0$. Moreover by constraints (5.13) we have that all vertices with $s_i^{v_1} = 0$ has at most one ingoing arc and the one vertex with $s_i^{v_1} = 1$ has at most ψ ingoing arcs. Therefore for all $k \in S$ and $l \in \mathbf{N} \setminus S$ it must holds that

$$\sum_{i,j \in S} y_{ij}^{v_1} \leq \sum_{h \in \mathbf{N}, g \in S \setminus \{k\}} y_{hg}^{v_1} + \psi s_k^{v_1} - \sum_{e \in \mathbf{N}} y_{el}^{v_1} + \psi s_l^{v_1} + 1 \tag{5.51}$$

Case 2 on S: Assume that there is a part of the route for v_1 outside of S and that v_1 is a *pseudo-simple* route. In this case clearly there must be at least one arc $y_{ij}^{v_1}$ where $i \in \mathbf{N} \setminus S, j \in S$.

$s_k^{v_1} = 0 \wedge s_l^{v_1} = 0$**:** Then $0 \leq \sum_{e \in \mathbf{N}} y_{el}^{v_1} \leq 1$. In this case the clover-cut holds if the following inequality holds $\sum_{i,j \in S} y_{ij}^{v_1} \leq \sum_{h \in \mathbf{N}, g \in S \setminus \{k\}} y_{hg}^{v_1}$. Since $s_k^{v_1} = 0$ and therefore there is at most one arc entering vertex k and since the whole route is not in S this inequality is trivially true for connected *pseudo-simple* routes.

$s_k^{v_1} = 1 \wedge s_l^{v_1} = 0$**:** The inequality becomes $\sum_{i,j\in S} y_{ij}^{v_1} \leq \sum_{h\in\mathbf{N},g\in S\setminus\{k\}} y_{hg}^{v_1} - \sum_{e\in\mathbf{N}} y_{el}^{v_1} + 1 + \psi$. In this case $0 \leq \sum_{e\in\mathbf{N}} y_{el}^{v_1} \leq 1$. Therefore it is enough to show that $\sum_{i,j\in S} y_{ij}^{v_1} \leq \sum_{h\in\mathbf{N},g\in S\setminus\{k\}} y_{hg}^{v_1} + \psi$ which clearly holds for a connected *pseudo-simple* route since $\psi \geq \sum_{h\in\mathbf{N}} y_{hk}^{v_1}$.

$s_k^{v_1} = 0 \wedge s_l^{v_1} = 1$**:** Since $\sum_{e\in\mathbf{N}} y_{el}^{v_1} \leq \psi$. Then clearly if $\sum_{i,j\in S} y_{ij}^{v_1} - 1 \leq \sum_{h\in\mathbf{N},g\in S\setminus\{k\}} y_{hg}^{v_1}$ the clover cut will hold. This is trivially true since $\sum_{h\in\mathbf{N},g\in S\setminus\{k\}} y_{hg}^{v_1} \geq \sum_{i,j\in S} y_{ij} + \sum_{m\in\mathbf{N}\setminus S,n\in S} y_{mn}^{v_1} - 1$.

Where $\sum_{m\in\mathbf{N}\setminus S,n\in S} y_{mn}^{v_1} \geq 0$ and thus $\sum_{h\in\mathbf{N},g\in S\setminus\{k\}} y_{hg}^{v_1} \geq \sum_{i,j\in S} y_{ij} - 1$.

Therefore this cut holds for all connected *pseudo-simple* paths.

Now we will prove that there exists S, k and l so that the clover-cut does not hold when v_1 is disconnected.

Let v_{1_1} and v_{1_2} be two disconnected components of v_1. Let S contain exactly the vertices of v_{1_1}. Clearly by constraints (5.12) v_{1_1} and v_{1_2} are cyclic. Since v_{1_2} is in $\mathbf{N}\setminus S$ and since $\mathbf{N}\setminus S \geq 2$ and $S \geq 2$, we can choose l on v_{1_2} so that $s_l^{v_1} = 0$ and k on v_{1_1} so that $s_k^{v_1} = 0$. By the choice of S there are no arcs entering S and therefore we have that $\sum_{i,j\in S} y_{ij}^{v_1} > \sum_{h\in\mathbf{N},g\in S\setminus\{k\}} y_{hg}^{v_1}$. Moreover as l is on v_{1_2} and $s_l^{v_1} = 0$ then $\sum_{e\in\mathbf{N}} y_{el}^{v_1} = 1$. Thus clearly $\sum_{i,j\in S} y_{ij}^{v_1} > \sum_{h\in\mathbf{N},g\in S\setminus\{k\}} y_{hg}^{v_1} + \psi s_k^{v_1} - \sum_{e\in\mathbf{N}} y_{el}^{v_1} + \psi s_l^{v_1} + 1$. □

For our case with *butterfly* routes $\psi = 1$ and the cut becomes:

$$\sum_{i,j\in S} y_{ij}^{v} \leq \sum_{h\in\mathbf{N},g\in S\setminus\{k\}} y_{hg}^{v} + s_k^{v} - \sum_{e\in\mathbf{N}} y_{el}^{v} + s_l^{v} + 1 \qquad v\in\mathbf{V}, \emptyset \subset S \subset \mathbf{N}, k\in S, l\in\mathbf{N}\setminus S \quad (5.52)$$

Initial Problem for the branch-and-cut Algorithm

When solving the problem using branch-and-cut the following relaxed problem is used as the initial problem to which the violated capacity and connectivity constraints are added.

$$LSNDPR = \textbf{Min:} \sum_{m\in\mathbf{M}} \sum_{(i,j)\in\mathbf{A}} c_{ij}^m \sum_{v\in\mathbf{V}} x_{ij}^{mv} + \sum_{m\in\mathbf{M}} \sum_{j\in\mathbf{N}} c_j^m \sum_{v\in\mathbf{V}} f_j^{mv} + \sum_{v\in\mathbf{V}} c^v h^v \quad (5.53)$$

s.t.

(Transhipment) $$f_i^{mv} \geq \sum_{j:(j,i)\in\mathbf{A}} x_{ji}^{mv} - \sum_{j:(i,j)\in\mathbf{A}} x_{ij}^{mv} \qquad m\in\mathbf{M}, i\in\mathbf{N}, v\in\mathbf{V} \quad (5.54)$$

(Flow) $$\sum_{v\in\mathbf{V}} \sum_{j:(i,j)\in\mathbf{A}} x_{ij}^{mv} - \sum_{v\in\mathbf{V}} \sum_{j:(j,i)\in\mathbf{A}} x_{ji}^{mv} = b_i^m \qquad i\in\mathbf{N}, m\in\mathbf{M} \quad (5.55)$$

(Cap 0) $$\sum_{m\in\mathbf{M}} x_{ij}^{mv} \leq y_{ij}^{v} M \qquad (i,j)\in\mathbf{A}, v\in\mathbf{V} \quad (5.56)$$

(Cap 2) $$r_{gh}^{v} + M(1 - y_{gh}^{v}) - q^v(t_{gh} + t_h) \geq 0 \qquad (g,h)\in\mathbf{A}, v\in\mathbf{V} \quad (5.57)$$

(route time) $$t_{max} - \sum_{(g,h)\in\mathbf{A}} r_{gh}^{v} \geq 0 \qquad v\in\mathbf{V} \quad (5.58)$$

(Cap 1) $$q^v C^v + M(1 - y_{ij}^{v}) \geq \sum_{m\in\mathbf{M}} x_{ij}^{mv} \qquad (i,j)\in\mathbf{A}, v\in\mathbf{V} \quad (5.59)$$

(Cyclic) $$\sum_{j:(i,j)\in\mathbf{A}} y_{ij}^{v} - \sum_{j:(j,i)\in\mathbf{A}} y_{ji}^{v} = 0 \qquad i\in\mathbf{N}, v\in\mathbf{V} \quad (5.60)$$

$$\text{(connected)} \quad \sum_{j:(i,j)\in\mathbf{A}} y_{ij}^v - s_i^v \leq 1 \qquad i \in \mathbf{N}, v \in \mathbf{V} \qquad (5.61)$$

$$\text{(Start vertex)} \quad \sum_{i\in\mathbf{N}} s_i^v = 1 \qquad v \in \mathbf{V} \qquad (5.62)$$

$$\text{(Ships)} \quad y_{ij}^v - h^v \leq 0 \qquad (i,j) \in \mathbf{A}, v \in \mathbf{V} \qquad (5.63)$$

$$y_{ij}^v \in \{0,1\} \qquad (i,j) \in \mathbf{A}, v \in \mathbf{V} \qquad (5.64)$$

$$x_{ij}^{mv} \geq 0 \qquad (i,j) \in \mathbf{A}, m \in \mathbf{M}, v \in \mathbf{V} \qquad (5.65)$$

$$f_i^{mv} \geq 0 \qquad m \in \mathbf{M}, i \in \mathbf{N}, v \in \mathbf{V} \qquad (5.66)$$

$$s_i^v \in \{0,1\} \qquad i \in \mathbf{N}, v \in \mathbf{V} \qquad (5.67)$$

$$q^v \geq 0 \qquad v \in \mathbf{V} \qquad (5.68)$$

$$r_{gh}^v \geq 0 \qquad (g,h) \in \mathbf{A}, v \in \mathbf{V} \qquad (5.69)$$

Note that the problem has been relaxed by removing the constraints:

(Butterfly-Cut:)

$$\sum_{i,j\in S} y_{ij}^v \leq \sum_{h\in\mathbf{N}, g\in S\setminus\{k\}} y_{hg}^v + s_k^v - \sum_{e\in\mathbf{N}} y_{el}^v + s_l^v + 1 \qquad v \in \mathbf{V}, \emptyset \subset S \subset \mathbf{N}, k \in S, l \in \mathbf{N}\setminus S$$

(Transhipment Cuts:)

$$u_{ji}^v + u_{ih}^v + 2(|T| - y^v(T)) \geq 2 \qquad (j,i),(i,h) \in T \in B(A), S((j,i),T) \neq S((i,h),T)$$

$$f_i^{mv} \geq \sum_{j,h\in\mathbf{N}, v\in\mathbf{V}} f_{jih}^{mv} - M(1 - s_i^v) \qquad m \in \mathbf{M}, i \in \mathbf{N}, v \in \mathbf{V}$$

$$f_{jih}^{mv} \geq x_{ji}^{mv} - x_{ih}^{mv} - M(2 - y_{ji}^v - y_{ih}^v + u_{ji}^v + u_{ih}^v) \qquad m \in \mathbf{M}, j,i,h \in \mathbf{N}, v \in \mathbf{V}$$

$$f_{jih}^{mv} \geq x_{ji}^{mv} - x_{ih}^{mv} - M(4 - u_{ji}^v - u_{ih}^v - y_{ji}^v - y_{ih}^v) \qquad m \in \mathbf{M}, j,i,h \in \mathbf{N}, v \in \mathbf{V}$$

Where $B(A)$ is the set of butterfly routes on the set of arcs A and T is a set of edges representing a butterfly route in $B(A)$. Let $S((j,i),T)$ be the simple tour of T on which arc (j,i) is located. Then, $S((j,i),T) \neq S((i,h),T)$ indicates that the arcs (j,i) and (i,h) are not on the same simple cycle of T. The cuts are added during the branching process on the initially relaxed variables y_{ij}^v and s_i^v. Clearly the violated cuts should be added as early as possible. However the cuts are defined for integer variables and for non-integer values the cuts may not be violated.

Connectivity cut separation algorithm

As mentioned earlier a cut is added when the corresponding constraint is violated. The cut added is the first found violated cut. To check if the connectivity is violated we can use depth first search to check if all ports assigned to a vessel can be reached from any other port assigned to the same vessel. The depth first algorithm needs to be run for each vessel and therefore has complexity $O(|V|(|N|+|A|))$. The variables y_{ij}^v are used to define a connection between two ports. Since the integrality of variables y_{ij}^v and s_i^v is relaxed the cut of the form:

$$\sum_{i,j\in S} y_{ij}^v \leq \sum_{h\in\mathbf{N}, g\in S\setminus\{k\}} y_{hg}^v + \psi s_k^v - \sum_{e\in\mathbf{N}} y_{el}^v + \psi s_l^v + 1 \qquad v \in \mathbf{V}, \emptyset \subset S \subset \mathbf{N}, k \in S, l \in \mathbf{N}\setminus S \quad (5.70)$$

may not be violated by the solution. Therefore additional conditions must be determined before adding the cut to the problem. The following conditions must hold when the connectivity cut (5.70) is violated for the integer relaxed problem.

- There are two vertices i and j visited by vessel v for which $y_{ij}^v = 0$.
- The flow on two disconnected components of a route when added must be greater than 1.

- There exists two disconnected vertices i and j visited by vessel v such that $s_i^v = 0$ and $s_j^v = 0$.

When all of the three conditions listed above are fulfilled a cut for every vessel and every $l \in T$ and $k \in S$ is added to the relaxed integer program.

Given a graph G, the separation algorithm is run for each vessel v as follows:

Connectivity-Separation-Algorithm(G, v)

1: $i \leftarrow$ a port with $y_{ij}^v > 0$;
2: $numberconnected \leftarrow DFS(i, v)$;
3: **if** $numberconnected$= ports on route for v **then**
4: **for all** ports i, j visited by v where $s_i^v = 0$ and $s_j^v = 0$ and $\sum_{h \in N} y_{ih}^v + \sum_{h \in N} y_{jh}^v > 1$ **do**
5: **for all** vessels v **do**
6: Add cut: $\sum_{g,h \in S} y_{gh}^v \leq \sum_{g \in \mathbf{N} h \in S \setminus \{i\}} y_{hg}^v + \psi s_i^v - \sum_{e \in \mathbf{N}} y_{ej}^v + \psi s_j^v + 1$
7: **end for**
8: **end for**
9: **end if**

Where $f(i) = \sum_{h \in N} y_{hi}^v$, for $i \in N$ is the flow on vessel v through the vertex i. The depth first search DFS in line 2 selects a vertex with an edge which has an edge weight greater than zero and uses the edges with edge weight greater than zero for the depth first search from this vertex. The DFS returns the number of ports the DFS has visited. The set of ports on route v is the ports with an in-edge with edge weight greater than zero.

Test results using this algorithm are reported in Section 7.7.

Transhipment cut separation algorithm

For finding transhipment cuts only *butterfly* routes are checked. For a *butterfly* route an arc leaving the center point is selected as start arc and by using this the last arc on the route is found. Then it is investigated if the difference in flow on the last and the first arc plus the difference of flow of the two other arcs at the centerpoint is greater than the the transhipment variable f_i^{mv}. If so the transhipment cuts are added. The transhipment cuts are of the form:

$$
\begin{aligned}
& u_{ji}^v + u_{ih}^v + 2(|T| - y^v(T)) = 2 \\
& f_i^{mv} \geq \sum_{j,h \in \mathbf{N}, v \in \mathbf{V}} f_{jih}^{mv} - M(1 - s_i^v) \\
& f_{hij}^{mv} \geq x_{ji}^{mv} - x_{ih}^{mv} - M(2 - y_{ji}^v - y_{ih}^v + u_{ji}^v + u_{ih}^v) \\
& f_{hij}^{mv} \geq x_{ji}^{mv} - x_{ih}^{mv} - M(4 - u_{ji}^v - u_{ih}^v - y_{ji}^v - y_{ih}^v)
\end{aligned}
$$

The set T must contain all arcs in the *butterfly* route. The cuts are only introduced if the solution is integer.

Given a graph G and a vessel v the capacity cut separation algorithm can be written as follows:

Transhipment-Separation-Algorithm(G, v)

1: $firstloop \leftarrow true$;
2: **for all** vertices i **do**
3: **if** $s_i^v \geq 1$ **then**
4: $centerpoint \leftarrow i$;
5: **end if**
6: **end for**
7: **if** less than 2 arcs leaving *centerpoint* used by vessel v **then**
8: **return** "No cut found";

9: **end if**
10: $startarc \leftarrow$ an arc (i, j) used by vessel v leaving $centerpoint = i$;
11: $Arc \leftarrow startarc$;
12: $T \leftarrow T \cup Arc$;
13: **while** Arc $(ij)^v$ exists $\wedge(j \neq centerpoint \vee firstloop)$ **do**
14: **if** $j = centerpoint$ **then**
15: $firstloop \leftarrow false$;
16: $flend \leftarrow Arc$;
17: $Arc \leftarrow$ arc leaving j used by vessel v which is not $startarc$;
18: $slstart \leftarrow Arc$;
19: **else**
20: $Arc \leftarrow$ arc used by vessel v leaving j;
21: **end if**
22: $T \leftarrow T \cup Arc$;
23: **end while**
24: **if** (Arc $(ij)^v$ exists) $\wedge(j = centerpoint) \wedge (\neg firstloop)$ **then**
25: $endarc \leftarrow$ Arc ;
26: **for all** $m \in \mathbf{M}$ **do**
27: **if** $f^{mv}_{centerpoint} < \max\{x^{vm}_{flend} - x^{vm}_{slstart}, 0\} + \max\{x^{vm}_{endarc} - x^{vm}_{startarc}, 0\}$ **then**
28: **return** $(startarc, endarc, slstart, flend, T)$;
29: **end if**
30: **end for**
31: **end if**
32: **return** "No cut found";

In the algorithm the start and the centerpoint of a *butterfly*-route is found at lines 2 to 6. It is checked at line 7 if the route of the vessel v is a *butterfly*-route. If it is not a *butterfly*-route then we will not need transhipment cuts. Then an arc exiting the start vertex is selected as the start arc for the route and the while loop walks through the route until all arcs have been visited and the last arc is then selected as the last arc on the route. In lines 24 to 32 it is investigated if there is a transhipment at the start vertex going between the two visits of vessel v. If this is the case then the cuts are added.

Note that the while loop is a depth first search and therefore the algorithm has complexity $O(|N|)$ for line 1 to 6 and $O(|\mathbf{A}|)$ for line 7 to 23 and $O(|\mathbf{M}|)$ for line 24 to 32. This gives a running time of $O(|\mathbf{N}| + |\mathbf{A}| + |\mathbf{M}|)$ for the algorithm. Note that if the route of vessel v is not a butterfly route the algorithm will terminate after an asymptotic running time of $O(|\mathbf{N}|)$.

5.5 Computational Experiments

The branch-and-cut algorithm was implemented in C++ using CPLEX version 10.2 and Concert version 2.4 where the Connectivity and Transhipment cuts were added when violated. This is compared with CPLEX version 10.2 MIP-solver on the compact model. Tests have been run on a dual Intel CPU with 2.67 GHz.

5.5.1 Test cases

We have constructed randomly generated cases that reflect real-life network design problems. The graphs have 5 to 15 ports and include up to 6 vessels. The forecast includes up to 12 demands.

In all of the test cases simple and *butterfly* routes are permitted, and the time of the evaluated period is 150 days. In the tests we use three different vessel types. Inspired by Agarwal and Ergun [1] the demands in the tests are selected randomly from the complete set of origin destination pairs, and the size of the demand is randomly selected between 1% and 80% of the capacity of the biggest vessel. The time of sailing between ports varies between 5 and 45 days. The cost of sailing

between two ports is dependent on the vessel type and the time it takes to sail the distance. The cost of transhipment at a port is randomly selected between two predefined extreme values. The ports are in all tests fully connected and the arcs are directed. The tests are terminated after 20 000 seconds which corresponds to 333.33 minutes. In order to ensure that the algorithms are tested without bias a new network is randomly generated for each test case.

5.5.2 Results

In Table 5.1 the test results for test cases with 5 and 7 ports are shown.

				Branch-and-cut				CPLEX	
				Cuts					
test	ports	vessels	demands	con	trans	Gap %	Time (min)	Gap %	Time (min)
a	5	3	9	6	140	0	0.05	0	0.22
b	5	3	9	13	196	0	0.13	0	0.33
c	5	3	9	16	252	0	0.08	0	0.23
d	5	3	9	21	364	0	0.15	0	0.40
e	5	3	9	11	252	0	0.16	0	0.87
a	7	3	9	29	560	0	15.28	0	46.91
b	7	3	9	29	476	0	0.93	0	3.36
c	7	3	9	30	504	0	0.21	0	5.99
d	7	3	9	17	280	0	0.53	0	3.73
e	7	3	9	67	1204	0	23.16	0	169.95
a	7	6	9	4	84	0	7.93	0	183.93
b	7	6	9	29	364	0	9.03	0.02	333.33
c	7	6	9	38	700	0	16.96	0.01	333.33
d	7	6	9	39	532	0	244.11	0.02	333.33
e	7	6	9	8	168	0	12.09	0.01	333.33
a	7	3	12	40	962	0	9.62	0	118.71
b	7	3	12	18	370	0	0.38	0	4.44
c	7	3	12	5	111	0	0.11	0	102.88
d	7	3	12	66	1628	0	24.70	0	166.88
e	7	3	12	26	740	0	0.41	0	5.48

Table 5.1: Test results with 5 to 7 ports and 9 to 12 demands given in the column of the same name. Comparing the described Branch and cut algorithm and the CPLEX branch and bound algorithm.

The first four columns in Table 5.1 indicate respectively the instance name, number of ports, vessels and demands in the test instance. The next four columns contain information related to solving the problem using branch-and-cut. The first two columns report the number of connectivity cuts added and the number of transhipment cuts added when solving the instance with the developed branch-and-cut algorithm. The next column, column 7 reports the gap between the upper and lower bound when the algorithm terminates. Column 8 reports time in minutes needed to solve the test instance. The next columns report the gap and the time in minutes needed to solve the test instance with CPLEX MIP-solver.

As in [1] the solution time is reported in minutes. It should be noted that it is not possible to directly compare our results to the results presented in [1] as the costs in the objective are different, the cost of transhipment is not included in [1], the solutions in [1] are heuristic, and we terminate the algorithms after 333.33 minutes whereas [1] allows the algorithm to run 1000 to 5000 minutes on instances with 20 ports. For instance, an instance with 6 ports and 6 demands, is in [1] covered by 30 vessels, where we focus on minimizing the number of vessels used.

All instances with 5 ports are in Table 5.1 solved to optimality within a minute, both using a branch-and-cut approach and CPLEX MIP-solver. However, the branch-and-cut algorithm terminates faster than the CPLEX MIP-solver. In fact for all tests cases in Table 5.1 the branch-and-cut algorithm outperforms the CPLEX MIP-solver, and CPLEX reaches the time out limit in four instances.

Table 5.2 reports the results for larger test cases with 10 to 15 ports. The columns are the same as in Table 5.1, except that two columns representing lower-bound (LB) and upper-bound (UB) for the branch-and-cut and for CPLEX are included. The bounds are reported, as some of the test instances are not solved to optimality. For each configuration of ports, ships and vessels two different instances are solved. The tests are stopped after 20 000 seconds and the best

				Branch-and-cut						CPLEX			
				Cuts					Time				Time
test	ports	v	dem	con	trans	LB	UB	Gap %	(min)	LB	UB	Gap %	(min)
a	10	3	7	32	462	60365	60365	0.00%	92.83	54818	61165	1.31%	333.33
b	10	3	7	44	484	50071	50071	0.00%	157.85	43487	50071	0.00%	333.33
a	10	6	9	212	3668	75296	100699	25.23%	333.33	74498	109882	31.48%	333.33
b	10	6	9	119	2352	68253	74155	7.96%	333.33	66213	85671	20.33%	333.33
a	15	3	7	8	22	50508	61305	17.61%	333.33	50505	*	*	333.33
b	15	3	7	101	1430	41457	49042	15.47%	333.33	40969	*	*	333.33
a	15	3	9	42	392	58874	78348	24.85%	333.33	58804	*	*	333.33
b	15	3	9	158	2604	63099	79249	20.38%	333.33	64130	*	*	333.33

Table 5.2: Test results with 10 to 15 ports. Comparing the described Branch and cut algorithm and the CPLEX branch and bound algorithm.

feasible solution found is reported. A gap between the best found feasible solution and the best known lower-bound is reported. When calculating the gap, the best known lower-bound is the maximum of the lower-bounds found by the branch-and-cut algorithm and the standard CPLEX MIP-solver. For all of the test cases with 15 ports, CPLEX was not able to find a feasible solution within the given time limit. For the cases reported in Table 5.2 the branch-and-cut algorithm in general performs at least as well as the CPLEX MIP-solver, both with respect to time and solution quality. However, for the very last instance listed in Table 5.2 with 15 ports the lower-bound of the CPLEX MIP-solver is slightly better than that of the branch-and-cut algorithm even though the CPLEX-MIP solver was not able to find a feasible solution. This may be due to the fact that CPLEX-MIP has all the transhipment and connectivity cuts included in the model, where the branch-and-cut approach gradually add the constraints, when violated. This results in higher lower bound early on, but results in a slower process of finding good feasible solutions. From the tests reported in Table 5.1 and 5.2 it is clear that increasing the number of ports considerably enhance the solution time of the problem. Therefore a time-space graph may not result in good solutions even though the connectivity cuts would be tightened.

The test results in general show improvements of the branch-and-cut scheme over CPLEX, and cases with 15 ports and a reasonable demand set can be solved to acceptable precision using this method. The problem size and solution time is, given the differences in problem structure, comparable to the tests reported in Agarwal and Ergun [1]. However, more vessels are needed in the solutions presented in [1] due to a difference in the objective function.

5.6 Conclusion

In this paper we have formulated a new model of the liner shipping network design problem. We have included *butterfly* routes in the route structure, included cost of transhipment in the objective and we have included the time of the route in the calculations of the capacity to present a realistic model. On the presented model we have generalized clover-cuts and transhipment cuts. We have described and implemented a separation algorithm for the branch-and-cut method. The proposed algorithm is so far the only exact solution method allowing transhipment and calculating the transhipment cost. To the best of our knowledge, branch-and-cut methods have not been applied to the liner shipping network design problem before. The test results show that the branch-and-cut method is promising for the liner shipping network design problem. Possible extensions of the model could be to include time windows for demand so that the real-life requirements for demand is investigated.

The contributions of this paper is a general formulation of the problem including transhipment and transhipment costs. Moreover, by applying the branch-and-cut method, we have developed an exact solution method which can solve instances of the same size as solved by some of the recently developed heuristic methods. The developed branch-and-cut method can be used for planning the routes of a smaller shipping company such as a feeder company or for planning a region of the network of a bigger liner shipping company. Since the algorithm finds optimal solutions it can

also be used to benchmark heuristic algorithms.

Acknowledgments

The authors wish to thank Christian Edinger Munk Plum, Berit Løfstedt, Shahin Gelareh, Jose Fernando Alvarez, Charlotte Vilhelmsen, Brian Kallehauge and two anonymous referees for valuable comments.

Bibliography

[1] R. Agarwal and O. Ergun. Ship scheduling and network design for cargo routing in liner shipping. *Transportation Science*, 42:175–196, 2008.

[2] J. F. Alvarez. Joint routing and deployment of a fleet of container vessels. *Maritime Economics & Logistics*, 11:186–208, 2009.

[3] N. Ascheuer, M. Jünger, and G. Reinelt. A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. *Computational Optimization and Applications*, 17:61–84, 2000.

[4] J. F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36:250–269, 2002.

[5] M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. In C. Barnhart and G. Laporte, editors, *Handbook in OR & MS*, volume 14, chapter 4. Elsevier Science B.V., 2007.

[6] M. Christiansen, K. Fagerholt, and D. Ronen. Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38:1–18, 2004.

[7] M. Christiansen and B. Nygreen. A method for solving ship routing problems with inventory constraints. *Annals of Operations Research*, 81:357–378, 1998.

[8] K. Fagerholt. Optimal fleet design in a ship routing problem. *International Transactions in Operational research*, 6:453–464, 1999.

[9] M. Fischetti, J. Salazar, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45:378–394, 1997.

[10] S. Gelareh and Q. Meng. A novel modeling approach for the fleet deployment problem within a short-term planning horizon. *Transportation Research Part E*, 46:76–89, 2010.

[11] M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32:263–273, 1998.

[12] B. Kallehauge, N. Boland, and O. B. G. Madsen. Path inequalities for the vehicle routing problem with time windows. *Networks*, 49:273–293, 2007.

[13] C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7:326–329, 1960.

[14] Mærsk. Mærsk line, services by map. World Wide Web electronic publication.

[15] T. Notteboom. The time factor in liner shipping services. *Maritime Economics & Logistics*, 8:19–39, 2006.

[16] T. Notteboom and J. Rodrigue. Containerisation, box logistics and global supply chains: The integration of ports and liner shipping networks. *Maritime Economics & Logistics*, 10:152–174, 2008.

[17] K. Rana and R. Vickson. Routing container ships using lagrangean relaxation and decomposition. *Transportation Science*, 25:201–214, 1991.

[18] D. Ronen. Cargo ships routing and scheduling: Survey of models and problems. *European Journal of Operational Research*, 12:119–126, 1983.

[19] D. Ronen. Ship scheduling: The last decade. *European Journal of Operational Research*, 71:325–333, 1993.

[20] K. Shintani, A. Imai, E. Nishimura, and S. Papadimitriou. The container shipping network design problem with empty container repositioning. *Transportation Research Part E*, 43:39–59, 2007.

[21] H. P. Williams. *Model Building in Mathematical Programming.* Wiley, New York, 1999.

Chapter 6

Dial-a-ride with synchronization for handicap assistance at airports

Line Blander Reinhardt* Tommy Clausen* David Pisinger*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
lbre@man.dtu.dk, tomc@man.dtu.dk, pisinger@man.dtu.dk

Abstract

The largest airports have a daily average throughput of more than 500 passengers with reduced mobility. The problem of transporting these passengers is in some cases a multi-modal transportation problem with synchronization constraints. A description of the problem together with a mathematical model is presented. The objective is to schedule as many of the passengers as possible, while ensuring a smooth transport with short waiting times. A simulated annealing based heuristic for solving the problem is presented. The algorithm makes use of an abstract representation of a candidate solution which in each step is transformed to an actual schedule by use of a greedy heuristic. Local search is performed on the abstract representation using advanced neighborhoods which modify large parts of the candidate solution. Computational results show that the algorithm is able to find good solutions within a couple of minutes, making the algorithm applicable for dynamic scheduling. Moreover high-quality solutions can be obtained by running the algorithm for 10 minutes.

6.1 Introduction

Around 1% of all passengers arriving at an airport need special assistance. Such passengers may be passengers returning from vacation with an injury, elderly or weak passengers, blind and deaf passengers, and passengers with other disabilities. We will refer to passengers needing assistance as passengers with reduced mobility (PRM). At the 31st biggest airport London Gatwick there was a throughput of 32 million passengers in 2009, of which around 900 each day needed assistance [10].

The support provided for the PRMs can be dedicated transport through the airport, and assistance at boarding. When assisting PRMs through an airport the PRM is picked up at the arriving location, e.g. check-in or gate of arrival, and delivered at the destination location, e.g. arrival hall or gate of departure.

Airports often have several terminals. At the studied airport the transport between the terminals is done in special buses solely for PRMs. Such buses will have a specific location for picking up

PRMs at each terminal. Moreover, for aircrafts not located at a gate, the PRM will be transported in a special bus between the gate and the aircraft. Therefore, the pickup and delivery of a PRM is represented as a number of pickup and delivery segments. The airport and airlines require that the PRMs are not left alone at any point during their journey through the airport, and the PRMs are required to be in their assigned flight seat at a fixed pre-specified time before departure. However, the PRM may be left alone for a while before boarding at the departing terminal in a supervised area. It may be possible to assist more than one PRM at a time depending on whether they are able to walk and orient themselves or the PRM is in a wheelchair. Each PRM is assigned a weight depending on their disability and the personnel and vehicles are assigned a capacity depending on their type.

Given a fixed set of transporters, the objective is to minimize the number of PRMs not delivered and to minimize the total unnecessary travel time used on the journeys. PRMs which cannot be delivered on time must be scheduled for a later flight. We view the problem of assisting PRMs as a dial-a-ride problem (DARP), which is a generalization of the pickup and delivery problem (PDP).

The DARP is NP-hard by reduction from the Hamiltonian cycle problem see Baugh et al. [1].

Between each delivery and pickup of a PRM the transporter delivering the PRM must meet the transporter picking up the PRM. This vehicle synchronization is called a temporal dependency, therefore the problem is a dial-a-ride problem with temporal dependencies (DARPTD). The concept of synchronization in routing was used by Ioachim et al. [11] for the fleet assignment problem and later extended to the more general temporal dependencies by Dohn et al. [9]. In pickup and delivery problems the similar problem of cross docking has been considered. In cross docking there is a transfer of goods between vehicles at the synchronized points. The pickup and delivery with cross docking is used in supply chain and planning city logistics systems [2], [7]. Pickup and delivery with cross docking was studied by Wen et al. [18] and Chen et al. [2]. In the cross docking problems the cross docking is optional for the vehicles. This is not the case in the problem of assisting PRMs at an airport, as the cross-docking points for each PRM are known and fixed. Moreover, even though cross docking problems often include a cost for the time the transshipped items spend at the cross docking facility, there is no requirement of synchronization. In the cross docking example considered by Chen et al. in [2] the demand is not a single pickup and delivery location pair. Instead, the demand is represented by a source and a sink for a given product and therefore the demand can be picked up at several different locations or delivered to several different locations. In the cross docking problems considered in [18] and [2] there will be at most one cross docking between the pickup and delivery of a resource. This also differs from the problem discussed in this paper as up to 4 synchronization points can be included in a transit journey through the airport. Other closely related problems are the pickup and delivery problem with transfers solved by Cortés et al. [6] in 2010 and the pickup and delivery with transfers and split delivery solved for liner shipping by Blander Reinhardt and Pisinger [16] in 2011. The PDP with transfers described in [6] contains a limited number of transfer points where the vehicles are synchronized. A model is presented and instances with up to 6 requests, 2 vehicles and 1 transfer point are solved to optimality using a combinatorial Benders decomposition method. The problem considered by Reinhardt and Pisinger in [16] does not consider synchronization at transfer points and is solved by branch and cut for instances a little larger than those of Cortes et al. [6].

The synchronization constraints and the objective also separate the transportation of PRMs in airports from the rich pickup and delivery problem described in [15]. In the survey by Cordeau and Laporte [3] from 2007 a list of some of the methods used for the dial-a-ride problem with multiple vehicles is provided. In this list the only exact methods are, a branch and cut method optimizing on vehicle travel cost by Cordeau [5], and an improvement on this method by Ropke et al. [17]. The exact method has been tested on a maximum 96 requests and 8 vehicles, which was solved in 71 minutes.

The dial-a-ride problems are usually solved by heuristics, as the studied problems often are real-life cases. Real life problems generally contain some additional constraints, which can be complicating and the objective varies. Moreover, in real-life, there can be constraints or desires not defined in the problem, and the size of the problems is often large. Due to this an optimal

solution may actually not be the best solution for the users.

Since the problem covered here is a dial-a-ride problem with complicating synchronization constraints and contains a large number of requests, vehicles and transfer locations, it is natural to consider heuristic solution methods. Moreover, there is a solution time requirement of 2 minutes given by the studied service provider. When solving instances with between 900 and 1500 requests within 2 minutes a heuristic method seems to be the only option.

For more details on the definition of DARP see Cordeau and Laporte [3] from 2007 and the more recent paper by Parragh et al. [14] where the variable neighborhood search heuristic is applied to a standard formulation of the dial-a-ride problem. Parragh et al. [14] report competitive solution for problems with up to 144 request based on the test sets delivered by Cordeau and Laporte [4]. Jaw et al. [12] in 1986 report finding good solutions to their dial-a-ride problem on an instance with 2617 requests and 28 vehicles using an insertion sort method. When run on present computers the method would satisfy the solution time requirement. Other heuristic methods, which are able to find good solutions for dial-a-ride problems with a large number of requests, are the regret insertion method by Diana and Dessouky [8] solving problems with 1000 requests, and a local search heuristic by Xiang et al. [19] solving problems with 2000 requests.

In this paper we present a local search heuristic for the specific problem based on simulated annealing. The algorithm makes use of an abstract representation, which is transformed to an actual schedule by use of a greedy heuristic. Local search is performed on the abstract representation using large neighborhoods. In each iteration, the resulting candidate solution is evaluated and accepted according to the standard criteria in simulated annealing. Computational results are reported showing that the algorithm is able to construct high-quality solutions in 10 minutes.

The main contribution of the paper is to present a highly relevant multi-modal transportation problem. With still more passengers traveling by air, we may expect increasing need for transport planning of passengers with reduced mobility. The problem is a true multi-modal transportation problem with synchronization, which may be used to model several other coupled pickup-and-delivery problems appearing in real-life problems. Finally, the proposed local search based on an abstract representation is generally applicable for other tightly constrained problems, and it shows promising results for the considered instances.

This paper is organized as follows. Section 6.2 contains a detailed problem description to ensure a thorough understanding of the operational process. In Section 6.3 a mathematical model of the problem is presented. Section 6.4 presents the solution method used. Section 6.5 contains the specifications of the data instances received. In Section 6.6 the tuning of the parameters for simulated annealing is described. Section 6.7 contains the results of the solution method applied to the real-life instances received. Finally in Section 6.8 the results and future work are discussed.

6.2 Problem Description

We will denote the considered problem the *Airport passenger with reduced mobility transport problem* (APRMTP). The APRMTP has been defined in cooperation with a service company providing the assistance for PRMs at a major transit airport. The company has 120 employees assisting between 300 and 500 PRMs through the airport each day. Each employee has a prespecified working area such as a specific terminal, driving between the terminals bus stop locations or driving between aircrafts and gates. An employee assigned to one area may not move into another area. Therefore, the journey of the PRM is split into a pickup and delivery for each of these areas. We call the pick up and delivery in a specific area for a *segment* and the ordered set of segments of a given PRM for a *journey*. The path of a bus or a foot personnel is referred to as *route*. On average there are three segments per PRM, and hence with 300 to 500 PRMs each shift we get a total of between 900 and 1500 pick up and delivery segments. This also includes assistance when boarding, which we have represented as a pickup and delivery request with special conditions. We will refer to the boarding assistance as *embarkment*. The employee assigned to an embarkment cannot go to another location between pickup and delivery of the embarkment segment. However, an employee may assist as many PRMs embarking on to the same flight as their capacity allows.

It should be noted that all of the pickup and delivery locations for every segment of the journey are predetermined.

As mentioned in the previous section, the PRM may not be left alone except at special supervised areas located in the departing terminal. This means that the employee delivering the PRM to a bus must wait with the PRM for the bus to arrive before being able to initiate the next task. The bus also has to wait for the employee to come and pick up the delivered PRM before continuing the route.

The company wants to make sure that they deliver the best service possible with the given number of employees and the current employee working area assignments. The PRMs are split into two categories: Those who are prebooked and those who are immediate. The prebooked PRMs order the service when purchasing the ticket or at least a few days in advance. Immediate PRMs request the service at check-in. If they are departing from the airport, the request is not known before the moment it has to be carried out. Immediate PRMs arriving on flights may be known down to half an hour before flight arrival. It is not always possible for the company to assist all PRMs and in such cases the prebooked PRMs have higher priority. The company also wishes to minimize unnecessary time the PRM spends on the journey segments. This could be time spent waiting to be picked up by the employee of the succeeding segment or extra travel time caused by picking up or delivering other PRMs before being delivered. Note, that the time spent at the supervised area of the departing terminal is not included in the service evaluation. The unnecessary time spent on the segments is called excess time. Clearly, when minimizing the overall traveling time there is a risk of having a few PRMs with very large traveling times. Therefore, it is important to limit the traveling times of the different segments so that the journey never becomes very unsatisfactory.

Many additional constraints concerning the pick-up and delivery times, assistance at embarkment, and transport to and from aircrafts not located at a gate, are imposed by the airport and airlines. Such constraints are listed in Table 6.1. The last two constraints in the table mean that embarkment cannot start later than 40 minutes before departure. When the aircraft is parked away from gate the embarkment must start even earlier than 40 minutes before departure.

	Additional constraints
1	An arriving PRM must be picked up exactly upon arrival
2	A terminal transfer is done by bus between the bus stop locations of the terminals
3	The PRM may not be left unsupervised
4	The PRM must not use more than 30 minutes of excess time on each segment
5	Embarkment lasts 20 minutes and cannot start earlier than 60 minutes before departure
6	The PRM must be seated in the aircraft exactly 20 minutes before departure

Table 6.1: List of constraints which are additional to the usual constraints in the standard dial-a-ride problem. The constraints often originate from a service contract agreement between the service provider, the airport and the airlines.

Note, that the person assisting the PRM through embarkment will not be able to leave the PRM until 20 minutes before departure if the aircraft is located at the gate, or before the PRM is picked up by a special vehicle if the aircraft is located away from the gate. In the latter case the special vehicle cannot leave the PRM before 20 minutes before departure.

The different transportation forms such as vehicles and assistance on foot have different capacities. Moreover the PRMs are assigned a volume depending on their disability. For example it is very hard for one person to push two wheelchairs, while two hearing or sight impaired persons easily can be assisted by the same employee.

6.2.1 Example of a journey of an PRM

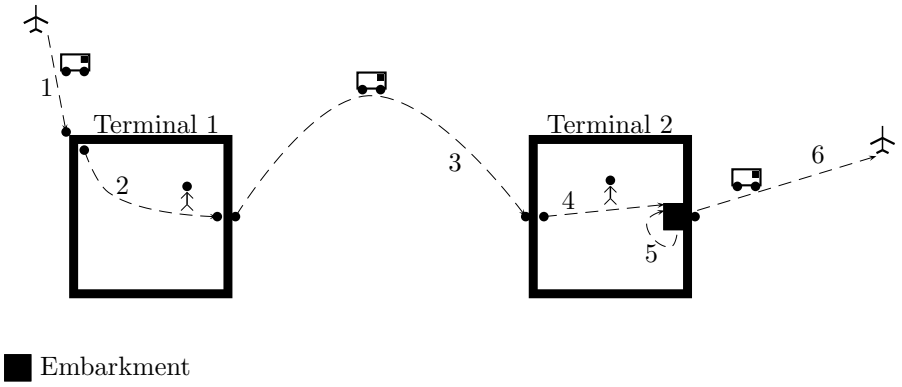


Figure 6.1: Illustration of a journey with six segments: (1) the PRM is picked up at arriving aircraft by vehicle at the exact time of arrival and delivered at gate, (2) the PRM is picked up at the gate by an employee. The bus in segment 1 cannot leave the gate before the PRM is picked up. The PRM is delivered at the bus stop for the inter-terminal busses, (3) The PRM is picked up by an inter-terminal bus and delivered to the bus stop at the terminal of the departing flight (the employee of segment 2 cannot leave before the bus arrives for pickup), (4) the PRM is picked up by an employee and delivered to the gate of the departing flight (the bus in segment 3 cannot leave before the employee arrives), (5) the PRM is assisted through embarkment at the gate. This task takes 20 minutes. For the time between segments 4 and 5 the PRM can be left at a special supervised lounge, (6) The PRM is picked up at the gate by a vehicle and delivered to the aircraft exactly 20 minutes before the departure time. The employee of segment 5 cannot leave before the vehicle arrives for pickup.

The most complex example of a journey is the case where a PRM makes a transit from an arriving aircraft not located at the gate to a departing aircraft in another terminal not located at the gate. Such a case is shown in Figure 6.1. We say that such a journey has six segments. All transit journeys are formulated as an ordered subset of these segments. Non-transit PRMs are either picked up or delivered to a public area in the terminal of their flight.

The algorithm presented in this paper is to be used in a dynamic setting, where immediate PRMs arrive continuously and disruptions in the daily plan such as flight delays frequently occur. Therefore, the company desires to receive a solution to the problem within a couple of minutes. We do not consider robustness and break times. However, robustness can be obtained by introducing buffer time in the time to get from one location to another and by altering the set of available employees. Breaks can be included by splitting the shift of an employee into several shifts.

6.3 Mathematical formulation

When describing the model it is important to bear in mind that the journey of each PRM is a set of pickup and deliveries called segments. This means that a PRM is picked up and delivered if all the segments of the journey are handled. Therefore, we must for each PRM ensure that all his/her segments are assigned before we consider the PRM delivered. As common for dial-a-ride problems with a heterogeneous fleet, each segment is represented by a pickup vertex and a delivery vertex specific to that segment. Also, there is a location inside each working area where the employees start and end their shift.

6.3.1 Graph representation

As mentioned earlier there is a vertex for each origin and destination of a segment and the location of all the vertices in a journey are predetermined. Since we already know which transport group is assigned to each vertex we can generate a disjoint graph for each transport group. Each graph has its own depot where the transporter start and end their shift. These graphs are "virtually" connected by the synchronization of the segments of a journey. For each terminal we have a directed graph connecting the vertices that must be serviced by foot personnel working in the given terminal. The busses transporting PRMs between terminals have a directed graph of their pickup and delivery vertices. The vehicles transporting PRMs from gates to aircrafts have a directed graph connecting their pickup and delivery vertices. The journey shown in Figure 6.1 has the segments 1,2,3,4 on four disjunct graphs. Both segment 4 and 5 are on the graph representing Terminal 2 and both segment 1 and 6 are on the graph representing vehicles operating between gates and aircrafts. Connections between pickup and delivery vertices, which are infeasible due to their time windows, are removed from the graph.

6.3.2 Mathematical model

Given the following sets:

K	The set of transporters. Contains all vehicles and persons on foot
R	The set of segments. Contains all the segments of all the journeys
R_p	A subset of segments in R which contains all the segments for PRM p
C	The set of all PRMs
B	The subset of C which contains all the prebooked PRMs
F	The set of departing flights
V^*	The set of pickup and delivery vertices and depots/bases
V	The subset of V^* containing all the pick up and delivery points/vertices
V_f	The subset of V containing all the embarkment vertices for flight $f \in F$
P	The set of pickup vertices, $P \subseteq V$ and P_p is the pickup vertices of PRM p
D	The set of delivery vertices, $D \subseteq V$ and D_p is the delivery vertices of PRM p
E	The set of edges connecting the elements in V^*
λ_p	The set (i, j) for a journey of PRM p where $i \in D_p$ and $j \in P_p$ are the delivery and pickup at a transfer point on the journey of p
δ	The set of vertex pairs that must be synchronized for handover

Each segment $r \in R$ has a start o_r and a destination d_r and the set V is all of the different o_r and d_r vertices. Each work area has a start point v_0 and an end point v_e representing the location, where the transporters start and end the day.

We define the following parameters:

M_b	The penalty for not transporting a prebooked PRM
M_n	The penalty for not transporting an immediate PRM
t_r	The minimum time needed to deliver segment $r \in R$
t_{ij}^k	The minimum time it takes to go from i to j on transport k
l'_j	The change in load at vertex $j \in V$
H	The maximum excess time allowed to be used on a segment
C_k	The capacity of transporter $k \in K$
M	A big constant being at least as large as the shift length
M_s	A big constant larger than the largest number of segments in any PRM
M_l	A big constant at least as large as the biggest capacity plus the largest volume possible for any PRM
a_i	The release time at vertex $i \in V^*$
b_i	The due time at vertex $i \in V^*$

We use the following variables:

s_i^k	the time when transporter k leaves vertex i
ϕ_p	An indicator variable indicating if a PRM $p \in C$ has a segment not assigned. ϕ_p is 0 if all segments of p are assigned and 1 otherwise
x_{ij}^k	An indicator variable indicating if the edge (i, j) is used by transporter k. x_{ij}^k is 1 if the edge is used by transporter k and 0 otherwise
l_i^k	the load on transporter k when leaving vertex i

As objective we have chosen a linear weighted combination of assigning as many PRMs as possible and minimizing the total excess time the PRMs spend on their journey. The model is:

$$\min \quad \sum_{d \in R} \left(\left(\sum_{k \in K} s_{d_r k} - s_{o_r}^k \right) - t_r \right) + M_b \sum_{p \in B} \phi_p + M_n \sum_{p \in C \setminus B} \phi_p \tag{6.1}$$

s.t.

$$\text{(P and D)} \quad \sum_{i \in V^*} x^k_{io_r} - \sum_{i \in V} x^k_{id_r} = 0 \qquad r \in R, k \in K \tag{6.2}$$

$$\text{(balance)} \quad \sum_{j \in V^*} x^k_{ij} - \sum_{j \in V*} x^k_{ji} = 0 \qquad i \in V,\, k \in K \tag{6.3}$$

$$\text{(start)} \quad \sum_{j \in V^*} x^k_{v_0 j} = 1 \qquad k \in K \tag{6.4}$$

$$\text{(end point)} \quad \sum_{j \in V^*} x^k_{j v_e} = 1 \qquad k \in K \tag{6.5}$$

$$\text{(P} \rightarrow \text{D)} \quad s^k_{d_r} - s^k_{o_r} \geq 0 \qquad k \in K,\, d \in R \tag{6.6}$$

$$\text{(Complete)} \quad M_s \phi_p - \sum_{d \in R_p} (1 - \sum_{k \in K} \sum_{j \in V} x^k_{o_r j}) \geq 0 \qquad p \in C \tag{6.7}$$

$$\text{(Timelimit)} \quad s^k_{d_r} - s^k_{o_r} - t^k_r \leq H \qquad k \in K,\, d \in R \tag{6.8}$$

$$\text{(Connect)} \quad s^k_i + t^k_{ij} + M(x^k_{ij} - 1) \leq s^k_j \qquad k \in K,\, (i,j) \in E \tag{6.9}$$

$$\text{(Handover)} \quad \sum_{k \in K} s^k_i = \sum_{k \in K} s^k_j \qquad (i,j) \in \delta \tag{6.10}$$

$$\text{(Journey)} \quad \sum_{k \in K} s^k_i \leq \sum_{k \in K} s^k_j \qquad (i,j) \in \lambda_p \tag{6.11}$$

$$\text{(Release)} \quad a_i \leq s^k_i + a_i(1 - \sum_{j \in V} x^k_{ij}) \qquad i \in V*,\, k \in K \tag{6.12}$$

$$\text{(Due)} \quad b_i \geq s^k_i + b_i(1 - \sum_{j \in V} x^k_{ji}) \qquad i \in V*,\, k \in K \tag{6.13}$$

$$\text{(Load)} \quad l^k_i + l'_j - M_l(1 - x^k_{ij}) \leq l^k_j \qquad (i,j) \in E,\, k \in K \tag{6.14}$$

$$\text{(Capacity)} \quad l^k_i \leq C_k \qquad i \in V,\, k \in K \tag{6.15}$$

$$\text{(Emb)} \quad \sum_{k \in K} x^k_{ij} = 0 \qquad j \in V \setminus V_f, i \in P \cap V_f, f \in F \tag{6.16}$$

$$\text{(Emb load)} \quad l^k_i - C_k(1 - \sum_{j \in V \setminus V_f} x^k_{ij}) \leq 0 \qquad k \in K, i \in D \cap V_f, f \in F \tag{6.17}$$

$$\text{(Variables)} \quad x^k_{ij} \in \{0,1\} \qquad k \in K,\, (i,j) \in E \tag{6.18}$$

$$\phi_p \in \{0,1\} \qquad p \in C \tag{6.19}$$

$$s^k_i \in \mathbb{R}^+_0 \qquad i \in V \cup \{p_k\},\, k \in K \tag{6.20}$$

$$l^k_i \in \mathbb{Z}^+_0 \qquad i \in V,\, k \in K \tag{6.21}$$

The objective function (6.1) sums all the excess time used on the segments and adds a penalty if a PRM is not delivered. Different penalties are used depending on whether PRM p is prebooked ($p \in B$) or immediate ($p \in C \setminus B$). Constraints (6.2) ensure that for each segment any PRM picked up is also delivered. Constraints (6.3) ensure that transporters leave all pickup or delivery vertices they enter. Constraints (6.4) ensure that all transporters leave their base. Constraints (6.5) ensure that all transporters return to their base. Constraints (6.6) ensure that on each segment a PRM is picked up before it is delivered. Constraints (6.7) ensure that any PRM with at least one segment not assigned generates exactly one penalty in the objective. Constraints (6.8) ensure that the excess time used on segment d does not exceed the maximum excess time H. Constraints (6.9) ensure that if an edge (i,j) is used then the time at which vertex j is visited is not earlier than the time leaving vertex i plus the travel time on edge (i,j). Constraints (6.10) ensure that the time of delivering transporter and pickup transporter are synchronized at their respective vertex points of $(i,j) \in \lambda$. Constraints (6.11) ensure that the segments of the journey are completed in the right order. Constraints (6.12) and (6.13) ensure that the segments are started and ended within their given time window. Constraints (6.14) update the load when a PRM is picked up or delivered. Note that since load is increased for any pickup vertex in V, constraints (6.14) together with constraints (6.9) ensure a connected route. This is true under the general assumption that

the transport from pickup to the delivery point is always greater than zero. Constraints (6.15) are the capacity constraints. Constraints (6.16) and (6.17) enforce the embarkment conditions of only starting embarkment tasks on the same flight before completing an embarkment segment. Constraints (6.16) only allow edges going from a pickup vertex of an embarkment segment to vertices of embarkment on the same flight. Constraints (6.17) ensure that when using an edge between an embarkment vertex and any vertex not belonging to an embarkment request for the same flight the load must be zero.

6.4 Solution method

The solution method we present is a greedy insertion heuristic combined with simulated annealing. The synchronization constraints present in the APRMTP add complexity to the generation of feasible solutions and the calculation of the objective value.

When a segment is inserted in a route of a bus or employee, it may influence not only the travel times of the later segments in the route inserted in, but also the other segments of the PRM and the segments of the routes of the personnel servicing those segments and so forth. This means that every time the end or start time of a segment is changed it may generate a cascade of changes on related segments. Therefore, when checking for feasibility one may, in worst case, have to evaluate the feasibility of all the segments in the problem. The same is true for calculating the objective as the insertion of a segment may influence the travel time on all remaining segments in all the routes. Therefore, an easy update of the objective when inserting a segment is not evident. However, together with a constraint on the maximum excess time allowed on each segment it may also constrain the problem significantly. An example of this is that the synchronization constraint reduces the number of possible feasible solutions.

We have constructed a greedy insertion heuristic for the initial solution, described in the next section, and later used in a simulated annealing scheme to improve solutions. The greedy insertion heuristic will, given an ordered list of PRMs, lead to a deterministic solution found with a search for best insertion spots, while the simulated annealing broadens the search by randomly selecting a neighborhood from a large neighborhood space and accepting solutions with a worse objective value. Due to the synchronization constraints the routes in the constructed solution are very interdependent and therefore it would be very difficult for a local search to find better solutions using neighborhood search. Instead of performing a local search on the constructed solution as done in GRASP we perform the local search on the abstract list representation of the previous solution.

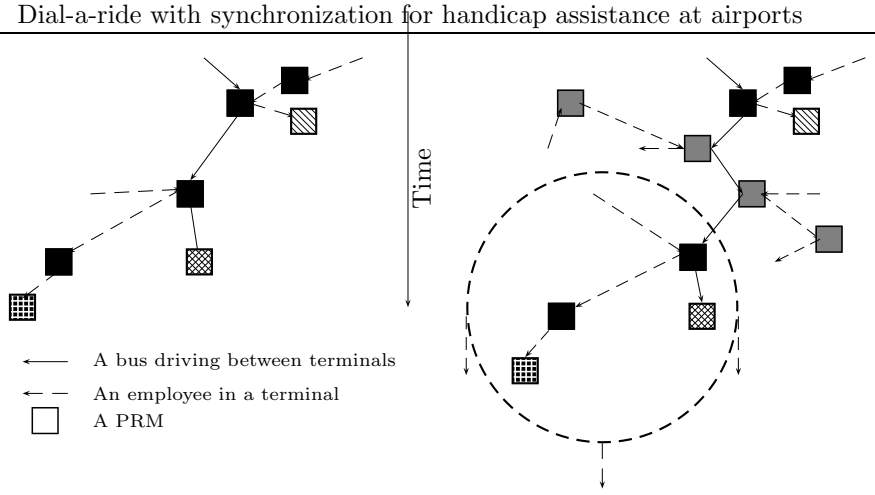
Figure 6.2 illustrates how PRM segments are moved when inserting a PRM segment into the route represented by the solid line.

6.4.1 Insertion heuristic

A greedy insertion heuristic (GIH) is created to quickly find a feasible solution. The heuristic takes two lists of PRMs, one containing the prebooked PRMs, and one containing the immediate PRMs. The insertion heuristic inserts first the PRMs from the prebooked list P_2, then the PRMs from the immediate list P_2 by going through the lists in sequential order. The reason for this is, that it is very important for the service provider to serve the prebooked PRMs.

We have sorted the lists by earliest pickup time of the PRM, starting at the earliest arrival time. For each PRM the segments are inserted in the order they appear in the journey and the next PRM is not inserted before all segments of the previous PRM are inserted.

For each segment the insertion place is found by only investigating the set of transporters working in the graph containing the given segment. The segment is inserted in the place and transporter where it creates the least increase in the total excess time. We only allow an insertion to push the time of the other segments forward. Therefore, when checking for feasibility, we only need to go through the segments with larger delivery times.



(a) Before insertion **(b)** After insertion of gray PRM

Figure 6.2: A section of the routes when inserting the gray PRM. The part of the graph inside the dashed circle is moved to a later time because of propagated delays caused by the insertion of the gray PRM. The patterned PRMs are the next PRMs to be handled by the employee or bus.

Usually, when minimizing the route cost, as in pickup and delivery problems, it is possible to calculate the new objective by the difference in the time introduced by the insertion and removal. However, in our case we did not do this as we found it too complicating when the excess time is minimized and the insertion can generate propagating delays induced by the synchronization constraints.

GIH(P_1, P_2)
1: **for** each PRM p taken first from list P_1 then from list P_2 **do**
2: **for** each segment s in p **do**
3: **for** each employee e serving s **do**
4: **for** each vertex v_1,v_2 in e in the possible time interval for s **do**
5: **if** load and time feasible insert s_{start} before v_1 and s_{end} before v_2 **then**
6: calculate total excess time for all inserted segments;
7: **end if**
8: **end for**
9: Select s_1 and s_2 where the least excess time is generated;
10: **end for**
11: **if** insertion was not possible **then**
12: Delete already inserted segments of p;
13: Register p as not inserted;
14: **else**
15: insert s in the e where the least excess time is generated;
16: **end if**
17: **end for**
18: **end for**

In the pseudo code of **GIH** the lines 1 and 2 are performed $O(n)$ times where $n = |R|$. The combinations that occur in line 3 and 4 can be $O(n^2)$. Checking for feasibility in line 5 is done by a depth first search which goes through $O(n)$ vertices updating their start/end times and the edge load. The calculation of total excess time of all inserted segments in line 6 is done by adding up excess time of all inserted segments, which is $O(n)$. Therefore, the asymptotic running time of the greedy insertion heuristic is $O(n^4)$.

In the test cases provided by the company the time windows are quite tight and by definition only a subset of employees are available in the area of a given segment. Therefore, the number of

feasible insertion points is limited before starting the propagated feasibility test and calculating the excess time.

6.4.2 Simulated annealing

A simulated annealing algorithm using the two lists of PRMs (prebooked and immediate) as an abstract representation was implemented. The initial solution is the greedy insertion heuristic on the two lists of PRMs, sorted by earliest possible start time. At each iteration of the simulated annealing algorithm, a number of moves takes place to obtain a candidate solution x from the current solution x'. The moves are made in the two PRM lists, which are then converted to a solution by the insertion heuristic. The moves are as follows:

- moving a not assigned PRM a random number of places forward in the respective list
- swapping the place of two PRMs randomly selected within the same list.

Note, that the prebooked PRMs and immediate PRMs will always remain in their respective lists. The lists are, when the moves are completed, converted by the greedy insertion heuristic into a schedule.

Let the objective function be defined as $f(x)$ for a solution x. If $f(x) \leq f(x')$ then x is accepted. Otherwise we accept the solution with probability:

$$\exp^{(f(x')-f(x))/T} \tag{6.22}$$

Details on the selected values for the temperature T will be covered in Section 6.6 on tunings. The temperature is decreased by a selected factor at each iteration. This decreasing factor is also called the cooling rate. The described large neighborhood and the acceptance probability allow the algorithm to escape local minima.

6.5 Data Instance and other parameter values

We have received 12 test cases from the service company covering dates September 20 to October 1, 2009. These test cases contain a total of 5000 PRM requests with information about the type and the position of the origin and destination. Travel time between the locations for the different transport forms were calculated from this information. Each test case represents a day and contains between 374 and 555 PRMs. The company delivers service in the majority of the aiport, but also other service providers are present in some parts of the airport. Therefore, the test cases do not cover all PRM requests at the airport.

Some of the PRMs in the data set were removed before running the tests due to corrupted data for the PRM or due to insurficient time available for the PRMs journey so that a solution transporting the PRM cannot exist satisfying the requirements given. This resulted in sets of between 353 and 495 PRMs. Each data set had a set of employees for the given day with a assigned terminal or vehicle. The number of employees assigned on a day was around 120. The employees were assigned to 6 different terminals and 2 different bus types. For each of the two bus types has specific area of operation.

The capacity of the transporters has been settled with the ground handling company as:

- An employee assisting inside terminal has capacity 2
- A bus between terminals has capacity 12
- A bus between gate and aircraft has capacity 9

For each PRM there is given a start time, an end time, a start location, an end location and a PRM type. There are 6 different types of PRMs in the data sets. For each types of PRM we have assigned a volume as follows:

WCHC Cannot walk or stand, needs wheel chair. Volume 1.5

WCHS Cannot walk up or down stairs. Volume 1

WCHR Cannot walk long distances. Volume 1

BLND Passenger is blind. Volume 1

DEAF Passenger is deaf. Volume 1

ASS Passenger cannot orient themselves. Volume 1

From the airport structure described to us by the company we have generated the segments for each PRM given the arrival and departure location of the PRM. For each shift between 900 and 1500 segments were generated. The excess time allowed for each segment is $H = 30$ minutes as this generally matches the requirement at airports. We have chosen the weights in the objective function as follows:

$$M_b = 1000, \qquad M_n = 400$$

This is to ensure a strong priority to the prebooked PRMs. Moreover, since a PRM journey has at most 6 segments, all PRMs must have an excess time of at most 180 minutes. Therefore, delivering any PRM will always be prioritized over a faster delivery and even several faster deliveries.

6.6 Tuning

The Simulated Annealing algorithm is tuned using 3 test cases selected from the received data sets. Theses instances represent the 20th of September (2009), 26th of September (2009) and the 1st of October (2009).

Since our insertion algorithm has a complexity of $O(n^4)$, the number of iterations completed in simulated annealing during the 2 minute running time is limited to a few hundreds. This means that for most instances the problem contains more PRMs than iterations performed during simulated annealing. Therefore, we consider the possibility for making several moves in each iteration. The moves are relocations in the list and do not influence the running time significantly. However, the neighborhood becomes much larger and the previous solutions may be ruined by a large number of moves.

We have included the initial objective value in the generation of the initial temperature to make the acceptance rate robust to variances in the size of the different problems. Thus the initial temperature is adjusted so that the probability of selecting a solution, which is exactly t percent greater than the initial solution is 50%. To find the best combination of the cooling rate and the number of moves, we have fixed the initial temperature so that a solution 5% worse than the initial solution must initially be accepted with 50% probability. This means that given an initial solution x and the temperature parameter of t, the initial temperature T is calculated as follows:

$$T = -tx/\log(0.5) \tag{6.23}$$

The algorithm is in Section 6.7 tested with the required solution time of two minutes, a slightly larger solution time of 5 minutes and a large solution time of 1 hour. To accommodate these tests the simulated annealing parameters are tuned both for a solution time of 2 minutes and a solution time of 1 hour. It is presumed that the values of the 5 minutes running time are close to the ones for 2 minutes and therefore we have not tuned for 5 minutes.

The algorithm is run ten times and the average solution of the ten runs is found. The average solution is then used to calculate the percent wise gap between the initial solution and the average solution found by simulated annealing. This is calculated as follows:

$$gap = \frac{initsolution - averagesolution}{initsolution} \cdot 100$$

Note, that by using the average solution the gap represents the expected improvement for a single run of the simulated annealing algorithm with the same solution time requirement.

We test a combination of different number of moves (4,12,20) and different cooling rates (0.5,0.8,0.9,0.95,0.99) given a fixed initial temperature T (using $t = 0.05$ in equation (6.23)). These tests are run with a time limit of respectively 2 minutes and for 1 hour for the 3 selected instances. The tuning results can be found in Appendix A.

For runs with time limit of 2 minutes, analyzing the gap of the test for combinations of cooling rate and number of moves, we have selected the value 0.9 for the cooling rate and 12 moves at each iteration. For runs with time limit of 1 hour, applying the same method, we have chosen the value 0.99 for the cooling rate and 4 moves at each iteration.

The selected values are used in the tuning tests on the initial temperature for 2 minutes and 1 hour time limit.

The initial temperature T is tested with the values of $t = \{0.01, 0.05, 0.1, 0.15\}$ in equation (6.23). For 2 minute runs the best solution is obtained when choosing the initial temperature so that a solution 5% greater than the initial value is accepted with 50% probability. These values will be used in Section 6.7 for tests with solution time requirement of 2 and 5 minutes. For 1 hour runs the best initial solution is obtained by choosing the initial temperature so that a solution 1% greater than the initial value is accepted with 50% probability. These values will be used for the tests in Section 6.7 with solution time requirement of 1 hour.

The values found in this section for the simulated annealing determines the intensification and diversification of the heuristic. The high number of moves at each iteration and the acceptance probability ensures diversification while the cooling rate gives rise to intensification over time. The start temperature determines the start diversification generated by the acceptance probability.

6.7 Test Results

The 12 test cases received from the service company covers the dates September 20 to October 1, 2009. Unfortunately, the company does not have records on how the PRMs actually were scheduled for the test cases. Hence, we cannot directly compare our schedules with the historic data as the service company does not currently make a plan of the day. Moreover from Cortes et al. [6] it can be seen that for a similar problem only tiny problems are so far solvable with exact method. In [6] problem sizes with up to 6 requests 1 transfer point and 2 vehicles is solved using both CPLEX and combinatorial Benders cut method.

It should be noted that in the test cases a little more than half of the PRMs are prebooked. The test cases are evaluated individually and therefore we include the 3 test cases used for tuning in this test section.

The tests were run on a computer with a 64 bit Intel Xeon 2.67 GHz CPU. The simulated annealing tests reported in this section with time requirement of 120 seconds and 300 seconds have been run with:

- Temperature: we use the factor $t = 0.05$ to adjust the temperature T according to equation (6.23)
- Cooling rate: 0.9
- Moves at each iteration: 12

The simulated annealing tests reported in this section with time requirement of 1 hour have been run with:

- Temperature: we use the factor $t = 0.01$ to adjust the temperature T according to (6.23).
- Cooling rate: 0.99
- Moves at each iteration: 4

Note, that the temperature is calculated given an initial solution x and the temperature parameter of t as described in equation (6.23).

In Table 6.2, we report the results of running the greedy insertion heuristic on the test case where the PRMs are sorted by earliest arrival time for all 12 data sets.

In column one the name of the data set is described by the date of the shift. We report how many PRMs in the given data set we had to remove due to corrupted data. The number of prebooked passengers not assigned (NAP) and not assigned immediate passengers (NAI) in the initial solution are reported in respectively column four and five. Finally, the initial solution retrieved from the greedy insertion heuristic on the PRM lists is reported in column six.

Case	prms	prms deleted	NAP	NAI	sol
20090920	374	21	0	4	3566
20090921	426	35	7	5	11430
20090922	451	23	0	5	5162
20090923	403	42	0	0	2001
20090924	465	33	0	3	4062
20090925	484	46	0	2	3500
20090926	401	27	0	0	1782
20090927	429	32	0	2	3095
20090928	401	23	0	0	2208
20090929	456	37	0	1	3259
20090930	555	60	7	9	13443
20091001	519	45	1	1	4401

Table 6.2: The results of the insertion heuristic run on the test cases. Note, that this is also the initial solution used by the simulated annealing heuristic.

From Table 6.2 it can be seen that no prebooked PRM were rejected for 9 of the 12 instances. The number of rejected immediate PRMs is also quite low compared to the number of PRMs when solving the problem using just the greedy insertion heuristic.

The results in Table 6.2 are used for evaluating the test results of the simulated annealing algorithm presented in Table 6.3.

In Table 6.3 the simulated annealing is tested for 120 seconds (2 minutes), 300 seconds (5 minutes) and 3600 seconds (1 hour) on each test case as given in column two. Each test is repeated ten times and the average value is reported. The temperature, cooling rate and number of moves are set to the selected values and the excess time allowed on each segment is set to $H = 30$ (minutes).

The average number of iterations and the standard deviation are reported in column three and four. In column five and six respectively, the average number of unassigned prebooked PRMs and unassigned immediate PRMs are reported. The best solutions of all the runs are reported in column seven and column eight contains the average solution. The gap between the average solution and the initial solution from Table 6.2 is reported in column nine. The results in Table 6.3 show that after running simulated annealing for 2 minutes the initial greedy heuristic solution is significantly improved. Increasing the solution time improves the solution quality in all cases and in some cases this improvement is significant. However, the improvements achieved from the initial solution in the first 5 minutes is in all cases greater than the improvements achieved in the following 55 minutes. The improvement achieved in the first 2 minutes from the initial solution is (in all cases except run 20090926) greater than the improvement achieved in the following 58 minutes. This indicates that the algorithm is good at finding improvements early in the algorithm and therefore works well with the requirement that a solution must be found within minutes.

Figure 6.3 and 6.4 show the improvement of solution values for a 2 minute run. It is seen that the algorithm is able to steadily improve the solution value.

The graphs in Figure 6.5 and 6.6 show the development in solution values for a single 1 hour run of respectively test instance 20090926 and 20090930. The figures show that after running simulated annealing for 15 minutes the improvements to the solutions become seldom and insignificant. This means that the significant reductions occur early in the simulated annealing and good solutions can be found after 10 minutes. In our case the solution time required is quite limited. In cases with not as tight running time requirements it would be relevant to test if multiple random restart

Case	time (s)	av it	stdD	av NAP	av NAI	best sol	av sol	gap
20090920	120	279.5	61	0	2.0	2423	2504.6	29.8%
20090920	300	767.6	52	0	2.0	2361	2426.9	31.9%
20090920	3600	8615.2	28	0	2.0	2110	2173.1	39.1%
20090921	120	209.0	369	5.6	3.6	8227	9026.0	21.0%
20090921	300	499.4	430	5.6	3.0	8112	8658.3	24.2%
20090921	3600	4813.4	390	5.1	2.4	7181	7475.3	34.6%
20090922	120	131.1	315	0	1.3	2922	3233.5	37.4%
20090922	300	317.8	85	0	1.0	2736	2838.9	45.0%
20090922	3600	3376	48	0	1.0	2222	2280.8	55.8%
20090923	120	244.0	50	0	0	1589	1659.4	17.1%
20090923	300	592.1	45	0	0	1464	1550.1	22.5%
20090923	3600	6567.3	15	0	0	1388	1409.9	29.5%
20090924	120	141.5	142	0	1.1	2670	2822.5	30.6%
20090924	300	309.7	85	0	1.0	2415	2578.8	36.5%
20090924	3600	3662.7	30	0	1.0	2037	2077.3	48.9%
20090925	120	150.4	60	0	0	1935	2051.8	41.4%
20090925	300	367.2	38	0	0	1910	1974.0	43.6%
20090925	3600	3999.3	17	0	0	1680	1714	51.0%
20090926	120	246.0	39	0	0	1422	1481.0	16.9%
20090926	300	597.3	53	0	0	1303	1351.9	21.1%
20090926	3600	6347.8	31	0	0	1072	1110.1	37.7%
20090927	120	212.7	34	0	0	1852	1899.4	38.6%
20090927	300	516.7	70	0	0	1672	1770.9	42.8%
20090927	3600	5987.0	56	0	0	1303	1396.4	54.9%
20090928	120	224.2	49	0	0	1533	1623.8	26.5%
20090928	300	544.8	46	0	0	1445	1512.9	31.5%
20090928	3600	5660.8	17	0	0	1243	1263.9	42.8%
20090929	120	123.8	85	0	0	2043	2221.3	31.8%
20090929	300	302.1	48	0	0	2003	2102.3	35.5%
20090929	3600	2937	73	0	0	1658	1752.3	46.2%
20090930	120	103.4	1478	2.8	5.0	4051	7351.9	45.3%
20090930	300	250.1	1264	1.3	3.3	3992	5142.8	61.7%
20090930	3600	2382.1	223	1.0	0.3	3130	3288.4	75.5%
20091001	120	112.1	168	0	0.3	2554	2777.1	36.9%
20091001	300	270.0	162	0	0.1	2230	2470.0	43.9%
20091001	3600	2537	57	0	0	1746	1848.1	58.0%

Table 6.3: The results of simulated annealing running for two minutes, five minutes and one hour.

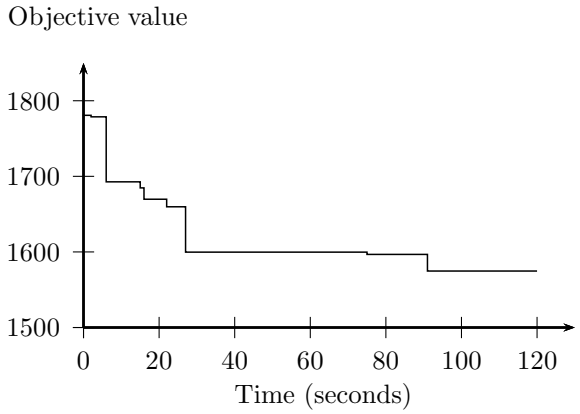


Figure 6.3: Development of solution values over time for test case 20090926

of the algorithm with a new ordering of the lists would allow for searches in other areas of the search space and possibly improve the solutions. In the current algorithm the neighborhood used is very large and therefore it is possible for the algorithm to investigate a larger area of the search space.

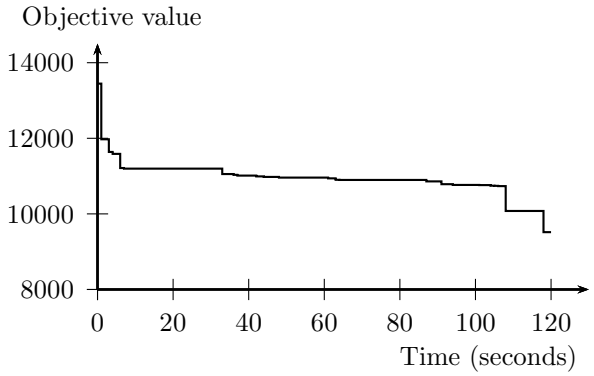


Figure 6.4: Development of solution values over time for test case 20090930

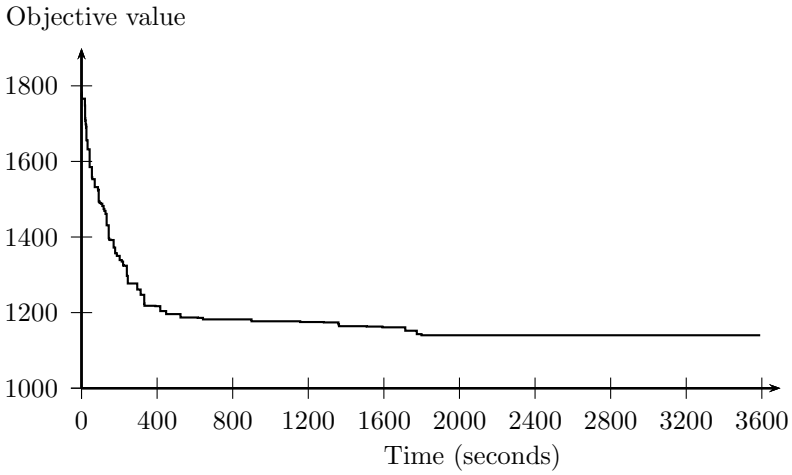


Figure 6.5: Development of solution values over time for test case 20090926

6.8 Conclusion

We have presented a model for the airport PRM transport problem and developed a heuristic with promising results, even when restricted to a running time of two minutes as required by the service company. Moreover, the number of rejected PRMs is very low also in the initial greedy insertion heuristic solutions.

Although the problem has been defined in cooperation with a specific handling company, we believe that the model is sufficiently general to cover most airports in the world. Moreover the model and solution method can easily be adapted to other multi-modal problems with many synchronization constraints.

The developed heuristic works well with the short solution time constraint, which can be seen by the fact that the big improvements are obtained within the first few minutes. The tests show that increasing the solution time slightly could in some cases give significant improvements. Such improvements could also be achieved by increasing computational power or algorithmic improvements such as parallelization of the program.

The next step in the academic area would be to test different solution methods on the problem to compare the solution quality and to reduce the computation time for the greedy insertion heuristic. One such paradigm could be to apply the combined MIP and ALNS heuristic presented in [13] which has shown good results for tightly constrained problems where even finding a feasible solution is an issue. In the application area the next step would be to incorporate this method at the users to see if the developed plans can improve their daily service.

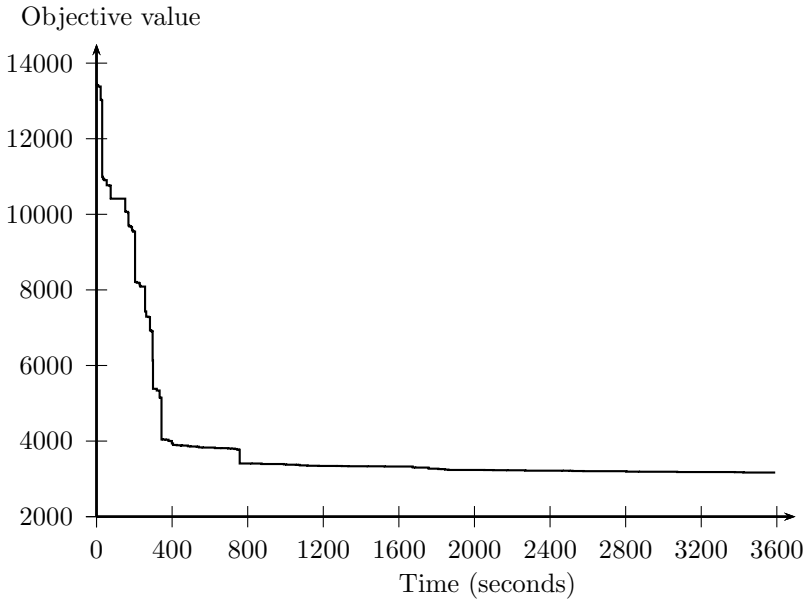


Figure 6.6: Development of solution values over time for test case 20090930

We have assumed that the personnel and the location of the personnel is fixed, but the short solution times make it possible to use local search to test whether relocation of some personnel may lead to a higher service level. Since the number of passengers arriving at the airport is increasing the heuristic can also be used to investigate when an increase in personnel is needed to deliver acceptable service to the increasing volumes of PRMs arriving.

Acknowledgments

The authors wish to thank Torben Barth, Berit Løfstedt, Christian Edinger Munk Plum, Charlotte Vilhelmsen and Mette Gamst for valuable comments. The authors also wish to thank Jacob Colding for presenting and clarifying the problem to us.

Bibliography

[1] J. W. Baugh, G. K. R. Kakivaya, and J. R. Stone. Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing. *Engineering Optimization*, 30:91–123, 1998.

[2] P. Chen, Y. Guo, A. Lim, and B. Rodrigues. Multiple crossdocks with inventory and time windows. *Computers & Operations Research*, 33:43–63, 2006.

[3] J. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153:29–46, 2007.

[4] J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579 – 594, 2003.

[5] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, 153:29–46, 2007.

[6] C. E. Cortés, M. Matamala, and C. Contardo. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research*, 200(3):711 – 724, 2010.

[7] T. G. Crainic, N. Ricciardi, and G. Storchi. Models for evaluating and planning city logistics systems. *Transportation Science*, 43:432–454, 2009.

[8] M. Diana and M. M. Dessouky. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B*, 38:539–557, 2004.

[9] A. Dohn, M. S. Rasmussen, and J. Larsen. The vehicle routing problem with time windows and temporal dependencies. Technical report, DTU Management, The Technical University of Denmark, 2009.

[10] A. Flower. Gatwick managing directors report. http://www2.westsussex.gov.uk/ds/cttee/gat/gat230709i7.pdf, 2009. publisher: www.gatwickairport.com.

[11] I. Ioachim, J. Desrosiers, F. Soumis, and N. Belanger. Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, 119:75–90, 1999.

[12] J. Jaw, A. Odoni, N. Psaraftis, and H. M. Nigel. A heuristic algorithm for the multi-vehicle advance request deal-a-ride problem with time windows. *Transportation Research part B*, 20:243–257, 1986.

[13] L. F. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. Submitted to European Journal of Oprational Research.

[14] S. Parragh, K. F. Doerner, and R. F. Hartl. A variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 37:1129–1138, 2010.

[15] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, 2007.

[16] L. Reinhardt and D. Pisinger. A branch and cut algorithm for the container shipping network design problem. *Flexible Services and Manufacturing Journal*, pages 1–26, 2011.

[17] S. Ropke, J.-F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49:258–272, 2007.

[18] M. Wen, J. Larsen, J.-F. Cordeau, and G. Laporte. Vehicle routing with cross-docking. *Journal of the Operational Research Society*, 60:1708–1718, 2009.

[19] Z. Xiang, C. Chu, and H. Chen. A fast heuristic for solving a large-scale dial-a-ride problem under complex constraints. *European Journal of Operational Research*, 174:1117–1139, 2006.

Appendix

A Appendix: Tuning tests

Table A1 describes the three test cases used for tuning. Using a time limit of 2 minutes, Table A2 reports results for tuning the coolrate and the number of moves per iteration, while Table A3 reports results for tuning the initial temperature parameter t. Using a time limit of 1 hour, Table A4 reports results for tuning the coolrate and the number of moves per iteration, while Table A5 reports results for tuning the initial temperature parameter t.

Case	PRMs	deleted	init NAP	init NAI	init sol
20090920	374	21	0	4	3566
20091001	519	45	1	1	4401
20090926	401	27	0	0	1782

Table A1: The test cases used for tuning, with the results from the greedy insertion heuristic. NAP stands for the number of not assigned prebooked PRMs and NAI stands for the number of not assigned immediate PRMs. The comn PRMs is the total number of PRMs provided in the instance of which the number given in 'deleted' is deleted

testcase	coolrate	moves	av ite	stdD	av NAP	av NAI	best sol	avg sol	gap
20090920	0.5	4	326.5	179	0	2.3	2491	2655.2	25.5%
	0.8	4	325.9	187	0	2.5	2457	2732.2	23.4%
	0.9	4	326.7	242	0	2.4	2426	2720.2	23.7%
	0.95	4	324.5	201	0	2.5	2511	2753.5	22.8%
	0.99	4	322.1	239	0	2.5	2605	2912.2	18.3%
	0.5	12	315.8	57	0	2.0	2396	2495.8	30.0%
	0.8	12	316.7	58	0	2.0	2396	2496.3	30.0%
	0.9	12	315.5	34	0	2.0	2461	2510.4	29.6%
	0.95	12	310.1	55	0	2.0	2431	2500.0	29.9%
	0.99	12	306.3	52	0	2.0	2541	2609.3	26.8%
	0.5	20	306.1	79	0	2.0	2351	2521.5	29.3%
	0.8	20	309.9	47	0	2.0	2418	2505.6	29.7%
	0.9	20	308.8	33	0	2.0	2450	2515.2	29.5%
	0.95	20	301.2	53	0	2.0	2437	2495.0	30.0%
	0.99	20	298.7	46	0	2.0	2464	2564.3	28.1%
20091001	0.5	4	125.3	198	0	0.4	2461	2691.7	38.8%
	0.8	4	125.4	130	0	0.8	2521	2796.7	36.5%
	0.9	4	123.2	120	0	0.5	2543	2794.8	36.5%
	0.95	4	120.0	187	0	0.2	2319	2737.5	37.8%
	0.99	4	117.7	205	0	0.7	2869	3149.8	28.4%
	0.5	12	120.8	111	0	0.3	2736	2838.9	35.5%
	0.8	12	115.1	161	0	0.1	2510	2742.0	37.7%
	0.9	12	114.4	190	0	0.3	2472	2723.3	38.1%
	0.95	12	107.7	126	0	0.3	2598	2749.2	37.5%
	0.99	12	104.1	130	0	0.2	2677	2893.6	34.3%
	0.5	20	113.2	131	0	0.1	2789	2891.6	34.3%
	0.8	20	110.4	141	0	0.2	2635	2834.0	35.6%
	0.9	20	106.1	184	0	0.3	2459	2820.9	35.9%
	0.95	20	103.3	136	0	0.4	2644	2831.5	35.7%
	0.99	20	100.0	129	0	0.6	2842	3048.1	28.1%
20090926	0.5	4	255.1	40	0	0	1378	1442.6	19.0%
	0.8	4	256.5	50	0	0	1368	1427.9	19.9%
	0.9	4	254.8	41	0	0	1352	1428.5	19.8%
	0.95	4	254.8	48	0	0	1395	1480.8	16.9%
	0.99	4	250.1	80	0	0	1437	1563.0	12.3%
	0.5	12	248.0	50	0	0	1425	1482.0	16.8%
	0.8	12	245.1	42	0	0	1420	1484.3	16.7%
	0.9	12	245.2	52	0	0	1322	1431.5	19.7%
	0.95	12	241.6	70	0	0	1380	1491.7	16.3%
	0.99	12	236.1	78	0	0	1399	1508.8	15.3%
	0.5	20	243.0	50	0	0	1400	1508.4	15.4%
	0.8	20	240.2	57	0	0	1424	1496.7	16.0%
	0.9	20	237.1	73	0	0	1370	1490.5	16.4%
	0.95	20	234.5	55	0	0	1347	1471.1	17.4%
	0.99	20	228.9	55	0	0	1422	1520.9	14.7%

Table A2: Tuning cooling rate and number of moves for solution time of 120 seconds

Case	temp.par. t	av it	stdD	av NAP	av NAI	best sol	av best sol	gap
20090920	1%	317.0	53	0	2.0	2379	2490.3	30.2%
	5%	315.5	34	0	2.0	2461	2510.4	29.6%
	10%	313.4	146	0	2.2	2454	2585.3	27.5%
	15%	312.6	36	0	2.0	2462	2542.9	28.7%
20091001	1%	107.0	175	0	0.4	2470	2812.2	36.1%
	5%	114.4	190	0	0.3	2472	2723.3	38.1%
	10%	109.2	130	0	0.2	2538	2717.9	38.2%
	15%	108.8	234	0	0.3	2428	2774.5	37.0%
20090926	1%	248.3	64	0	0	1387	1470.3	17.5%
	5%	245.2	52	0	0	1322	1431.5	19.7%
	10%	241.2	54	0	0	1370	1473.8	17.3%
	15%	242.0	44	0	0	1384	1486.4	16.6%

Table A3: Tuning initial temperature parameter t given required solution time of 120 seconds, 12 moves and cooling rate of 0.9.

testcase	coolrate	moves	av ite	stdD	av NAP	av NAI	best sol	avg sol	gap
	0.5	4	7706.9	36	0	2.0	2170	2240.7	37.2%
	0.8	4	8129.1	29	0	2.0	2146	2200.0	38.3%
	0.9	4	8985.3	44	0	2.0	2089	2164.4	39.3%
	0.95	4	8968.7	27	0	2.0	2146	2194.1	38.5%
	0.99	4	8957.7	27	0	2.0	2121	2177.2	38.9%
	0.5	12	7854.3	19	0	2.0	2197	2242.2	37.1%
	0.8	12	8793.4	30	0	2.0	2166	2207.8	38.1%
20090920	0.9	12	8762.2	24	0	2.0	2165	2202.5	38.2%
	0.95	12	7270.6	26	0	2.0	2165	2200.8	38.3%
	0.99	12	7309.2	28	0	2.0	2160	2204.4	38.2%
	0.5	20	8863	29	0	2.0	2212	2261.2	36.6%
	0.8	20	6721.1	33	0	2.0	2250	2297.9	35.6%
	0.9	20	5958.8	28	0	2.0	2221	2261.5	36.6%
	0.95	20	5939.4	38	0	2.0	2221	2280.7	36.0%
	0.99	20	8281.6	35	0	2.0	2200	2275.3	36.2%
	0.5	4	3284.7	61	0	0	1723	1848.4	58.0%
	0.8	4	3218.4	97	0	0	1689	1844.8	58.1%
	0.9	4	3234.7	57	0	0	1723	1823.7	58.6%
	0.95	4	3150.8	104	0	0.1	1653	1849.6	58.0%
	0.99	4	3010.5	64	0	0	1685	1782.5	59.5%
	0.5	12	3166.4	170	0	0	1794	2083.4	52.7%
	0.8	12	3028.0	139	0	0.1	1800	2037.6	53.7%
20091001	0.9	12	2219.3	77	0	0	1972	2118.8	51.9%
	0.95	12	2228.1	58	0	0	1949	2051.5	53.4%
	0.99	12	2434.8	92	0	0	1943	2043.6	53.6%
	0.5	20	3043.4	93	0	0	2035	2211.1	49.8%
	0.8	20	2582.9	102	0	0	2055	2223.2	49.5%
	0.9	20	1909.7	138	0	0	2115	2284.2	48.0%
	0.95	20	2089.4	66	0	0	2136	2219.3	49.6%
	0.99	20	1778.5	66	0	0	2068	2158.8	50.9%
	0.5	4	5982.2	22	0	0	1093	1117.8	37.3%
	0.8	4	3916.9	16	0	0	1050	1083.8	39.2%
	0.9	4	4712.9	35	0	0	1035	1092.8	38.7%
	0.95	4	6418.6	17	0	0	1043	1066.9	40.1%
	0.99	4	4122.6	38	0	0	1032	1077.9	39.5%
	0.5	12	6815.5	35	0	0	1094	1139.4	36.1%
	0.8	12	4903.2	32	0	0	1086	1147.0	35.6%
20090926	0.9	12	4479.0	53	0	0	1078	1154.6	35.2%
	0.95	12	5900.2	25	0	0	1095	1138.9	36.1%
	0.99	12	6127.5	39	0	0	1128	1162.3	34.8%
	0.5	20	5755.6	28	0	0	1171	1203.7	32.5%
	0.8	20	5968.6	26	0	0	1177	1221.8	31.4%
	0.9	20	6252.7	44	0	0	1145	1222.3	31.4%
	0.95	20	6511.7	44	0	0	1167	1211.5	32.0%
	0.99	20	5926.8	39	0	0	1149	1187.5	33.4%

Table A4: Tuning cooling rate and number of moves for solution time of 1 hour

Case	temp.par.t	av it	stdD	av NAP	av NAI	best sol	av best sol	gap
	1%	8821.1	30	0	2.0	2123	2188.4	38.6%
20090920	5%	8769.1	23	0	2.0	2166	2190.7	38.6%
	10%	8694.1	32	0	2.0	2113	2181.9	38.8%
	15%	8287.5	40	0	2.0	2148	2206.5	38.1%
	1%	3097.5	72	0	0	1632	1759.6	60,0%
20091001	5%	2916.4	128	0	0	1578	1790.0	59.3%
	10%	2470.2	87	0	0	1697	1828.2	58.5%
	15%	2758.6	71	0	0	1750	1856.9	57.8%
	1%	6256.2	23	0	0	1048	1082.5	39.3%
20090926	5%	6237.9	21	0	0	1045	1077.4	39.5%
	10%	6378.6	27	0	0	1069	1108.6	37.8%
	15%	6617.1	33	0	0	1037	1089.4	38.9%

Table A5: Tuning initial temperature given required solution time of 1 hour, 4 moves and cooling rate of 0.99

Chapter 7

The vehicle routing problem with edge set costs

Line Blander Reinhardt* Mads Kehlet Jepsen* David Pisinger*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
lbre@man.dtu.dk, pisinger@man.dtu.dk, makj@man.dtu.dk

Abstract

We consider an important generalization of the vehicle routing problem with time windows in which a fixed cost must be paid for accessing a set of edges. This fixed cost could reflect payment for toll roads, investment in new facilities, the need for certifications and other costly investments. The certifications and contributions impose a cost for the company while they also give unlimited usage of a set of roads to all vehicles belonging to the company. Different versions for defining the edge sets are discussed and formulated. A MIP-formulation of the problem is presented, and a solution method based on branch-and-price-and-cut is applied to the problem. The computational results show that instances with up to 50 customers can be solved in reasonable time, and that the branch-cut-and-price algorithm generally outperforms CPLEX. It also seems that instances get more difficult when the penalized edge sets form a spanning tree, compared to when they are randomly scattered.

7.1 Introduction

In certain real-life situations the cost of a connection does not entirely depend on the cost of the individual links (edges). Frequently, in real life, a fee must be paid by the company for allowing its vehicle to access roads, areas, bridges or other. Such a fee may in some cases only be required to be paid once by the company and is in such cases independent of the number of vehicles accessing any edge in the set. Companies routing in an area with many ferry connections may pay to access a set of ferries owned by a company at a monthly rate or at a reduced price. Here, it is important to determine which ferry companies it is most profitable to use. The same applies to Toll roads and bridges, where some countries charge a company based tax for accessing all freeways in the country (see Figure 7.1). In war zones or areas of unrest a company may need to get a certification allowing its vehicles to travel on certain protected roads or to enter certain protected zones. Even though in some cases the access is to be paid only for the vehicle accessing the edge set the company will often wish to sign up all its vehicles for robustness and easy administration purposes. Yet another situation where a set of edges can be accessed at a fixed cost is in cases where there is an option of investing in a facility. In such problems, referred to as location-routing problems, there is often a fixed charge connected to a facility and location. Nagy and Salhi [20] give an

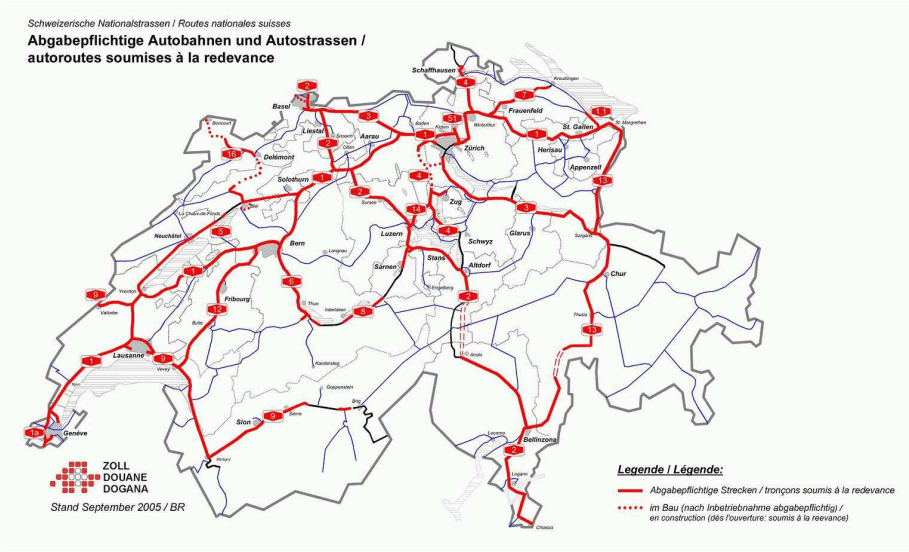


Figure 7.1: Main road net in Switzerland. To access all the red edges, a vignette needs to be paid. A transportation company may choose to avoid the toll roads and only use the ordinary highways

extensive survey of location-routing problems covering many different routing problems combined with facility location problems. Belenguer et al. [2] recently presented a branch-and-cut method for the location routing problem which apart from considering multiple depots, can be seen as a special case of the model presented in this paper. In all of the mentioned cases there is a fixed charge for accessing a set of edges. For the case shown in Figure 7.1 there may be both an edge belonging to an edge set and an edge with no additional cost connecting the same pair of vertices. A graph where multiple edges exists between two vertices is called a multi-graph.

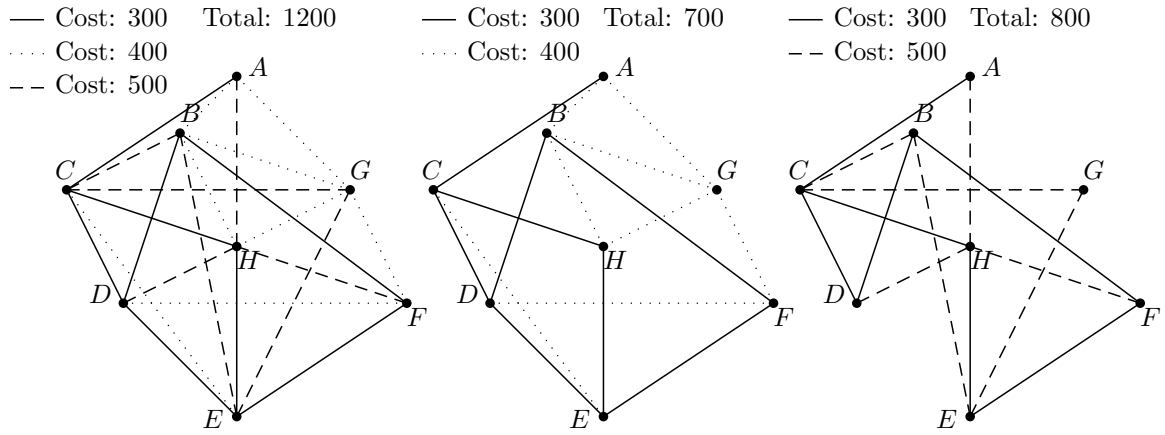
The CPLEX modeling of a multi-graph with edge set cost is considered in Section 7.5.2 however a branch-cut-and-price algorithm for the multigraph with edge set cost has not been developed for this paper.

The problem of minimizing the overall cost when planning routes that have a cost associated with sets of edges is in this paper investigated as a generalization of the well known problem of routing vehicles with capacity and service time window restrictions (VRPTW).

In the version of the VRPTW considered here the edges of the graph belong to different sets. Once the cost of accessing the set is paid all vehicles can access the edges in the set. However, there might still be a price associated with each of the edges used. Note that the price for accessing the set is paid at most once. This cost has an influence on all the routes since once the access price is paid the edges can be accessed by another vehicle without paying the access price again. This makes the cost of the different vehicle routes interdependent. We will denote the considered problem an *edge set vehicle routing problem with time windows* (ESVRPTW). In Figure 7.1 an example of a network with the edges partitioned into different sets is shown. Figure 7.1 a) shows the entire set of edges, and b) and c) show accessible edges when paying for different combinations of two edge set. Clearly the ESVRPTW is NP-hard as it is a generalization of the VRPTW. Eventhough edges refers to bidirectional links and directional arcs generally are used for the VRPTW this does not change the problem as the arcs representing the two directions of the edge are both assigned to the same edge set.

We will in this paper present a model for the problem and a Danzig-Wolfe decomposition similar to the classical decomposition of the VRPTW.

The paper is organized as follows. In the following section we give a rough overview of literature for the vehicle routing problem with time windows and describe relevant results that can be used for



(a) All 3 edge sets of the graph. **(b)** Accessible edges when the two edge sets shown are paid for. **(c)** Accessible edges when the two edge sets shown are paid for.

Figure 7.1: The graph of a) all three edge sets b) two edge sets and c) another two edge sets

solving the ESVRPTW. Section 7.3 presents a MIP model for the ESVRPTW and in Section 7.4 the decomposition of the problem into a Master and subproblem is described. Moreover the solution method and valid inequalities are described. In Section 7.5 various extensions of the ESVRPTW model are discussed. In Section 7.6 the test instances are described. Section 7.7 reports computational results, and finally the paper is concluded in Section 7.8.

7.2 Literature review

To the best of our knowledge, the problem of routing vehicles with an edge set cost has not yet been investigated in the published literature. However, the underlying problem, the vehicle routing problem with time windows (VRPTW), has been extensively studied. The vehicle routing problem was introduced in 1959 by Dantzig and Ramser in [6] as the truck dispatching problem. Many different exact and heuristic methods have been applied to the problem. The basis of the research in this paper is in the exact methods. In 1981 Christofides et al. [4] presented a decomposition generating q-routes for the capacitated VRP. One of the first exact methods for the VRPTW was by Kolen et al. [16] using the ideas presented in [4] and applying them to the VRPTW. This was later included in a branch-and-price method by Desrochers et al [9].

In 1987 a benchmark suite was presented for the VRPTW [22] making it easy to compare solution methods and the research society has been enticed by the problem of solving these tests. Recently there has been a strong development in solution times and problem sizes solved to optimality. In 1999 Kohl et al. [15] and Cook and Rich [5] both applied branch-cut-and-price to the VRPTW.

Some of the most recent developments in solving the VRPTW are described in [1], [8],[12], and [14]. Both Jepsen et al. [12] and Baldacci et al. [1] use the valid cuts suggested by Lysgaard et al. [19] to separate candidate sets for branching. Even though the cuts in [19] are implemented for the capacitated vehicle routing problem (CVRP) they may be used for the VRPTW, as the solutions to the VRPTW are a subset of the solution to the corresponding CVRP. Jepsen et al. [12] implemented a branch cut and price algorithm with a label-setting bi-directional algorithm for elementary shortest paths developed by Righini and Salani [21]. Jepsen et al. added the subset-row (SR) inequalities on the master problem variables and modified the subproblem to include the reduced cost from these inequalities. These SR inequalities are included by both Desaulniers et al. [8] and Baldacci et al. [1] in their algorithms.

Desaulniers et al. in 2008 [8] further improved the results by using Tabu search for finding

improving routes in the subproblem and generalized the k-path inequalities originally formulated by Kohl et al. [15].

Baldacci et al. [1] introduce an enumeration framework. The master problem is solved using a subgradient optimization algorithm and enumeration is done by solving an ESPPRC where standard dominance is limited. To improve the dominance a lower bound for the completion of each label is found using the ng-routes.

In this paper we formulate the ESVRPTW and solve it using the solution method used by Jepsen et al. [12] on the VRPTW as this method can be easily adapted to solve the ESVRPTW.

7.3 The Model

The mathematical model is based on the model presented in [12]. Given the following sets:

C	The set of customers
R	The set of edge groups
V	The set vertices representing the customers in C and the depot defined as 0
$\mathbf{A}$	The set of arcs (i, j) in V and $\mathbf{A}_r$ is the set of arcs (i, j) belonging to the group $r \in R$
K	The set of vehicles, where $\|K\|$ is unbounded according to the standard VRPTW definition.

The variables are defined as:

x_{ij}^v	Indicator variable indicating if the arc (i, j) is used by vehicle $v \in K$
y_r	Indicator variable which is one if an edge from group $r \in R$ is used and zero otherwise
t_i^v	The time vehicle v visits $i \in V$.

The parameters are defined as:

D	The capacity of the vehicles
d_i	The demand which must be delivered to vertex $i \in V$. The demand at the depot is zero
a_i	The availability time for customer $i \in C$. Note that $a_i \geq 0$.
b_i	The required completion time for customer $i \in C$ with $b_i \geq a_i$
θ_{ij}	The time it takes to travel from $i \in C$ to $j \in C$ on arc (i, j).
c_{ij}	The cost of using an arc $(i, j) \in A$
c_r	The cost of accessing the arcs in group $r \in R$

Since the problem is a generalization of the VRPTW the model presented here for the ESVRPTW is the standard VRPTW model presented by Kallehauge in the survey [13], with an additional set of constraints used to formulate the edge set costs. The cost of the edge sets are inserted into the objective. In the presented model the assumption is that each edge belongs to exactly one set, however, alternatives to this assumption are discussed in Section 7.5.

$$\textbf{Min:} \sum_{v \in K} \sum_{(i,j) \in \mathbf{A}} c_{ij} x_{ij}^v + \sum_{r \in R} c_r y_r \tag{7.1}$$

$$s.t. \quad y_r - \sum_{v \in K} x_{ij}^v \geq 0 \qquad \forall r \in R,\ (i,j) \in \mathbf{A}_r \tag{7.2}$$

$$\sum_{v \in K} \sum_{(i,j) \in \mathbf{A}} x_{ij}^v = 1 \qquad \forall i \in C \tag{7.3}$$

$$\sum_{i \in C} x_{i0}^v = \sum_{i \in C} x_{0i}^v \qquad \forall v \in K \tag{7.4}$$

$$\sum_{(j,i) \in \mathbf{A}} x_{ji}^v - \sum_{(i,j) \in \mathbf{A}} x_{ij}^v = 0 \qquad \forall i \in C, \forall v \in K \tag{7.5}$$

$$\sum_{(ij) \in \mathbf{A}} d_i x_{ij}^v \leq D \qquad \forall v \in K \tag{7.6}$$

$$a_i \leq t_i^v \leq b_i \qquad \forall\, i \in V, v \in K \tag{7.7}$$

$$(t_i^v + \theta_{ij}) x_{ij}^v - t_j^v \leq 0 \qquad \forall\ v \in K,\ (i,j) \in \mathbf{A} \tag{7.8}$$

$$x_{ij}^v \in \{0,1\} \qquad \forall\, (i,j) \in \mathbf{A},\ v \in K \tag{7.9}$$

$$y_r \in \{0,1\} \qquad \forall\ r \in R \tag{7.10}$$

$$t_i^v \in \mathbb{R}_0^+ \qquad \forall\, i \in V,\ v \in K \tag{7.11}$$

The objective (9.1) is the sum of the cost on the edges accessed and the sum of the cost of accessing the sets of the edges accessed. The constraints (7.2) ensure that if an edge in a set is used then the cost of accessing the set is paid. Note that the integrality of the x_{ij}^v variables implies integral y_r variables. Constraints (7.3) ensure that every customer is visited. Constraints (7.4) ensure that all vehicles start and end their journey at the depot. Constraints (7.5) ensure that vehicles arriving at a customer also leaves the same customer. Constraints (7.6) ensure that the capacity of a vehicle is not exceeded. Constraints (7.7) ensure that customers are visited in their respective time window. Finally, constraints (7.8) finds the time of vehicle v at customer i. If vehicle v does not visit customer i any time can be chosen. Constraints (7.8) also ensures that the route is simple. The variables x_{ij}^v and y_r are in (7.9) and (7.10) defined to be binary and the variable t_i^v is in (7.11) defined to be a positive real number.

7.3.1 Tightening of the edge set constraints

In the ESVRPTW each customer must be visited exactly once. This requirement is ensured by constraints (7.3) and can be used to tighten the constraints in (7.2). Since each customer is visited once, we know that if several edges belonging to the same set leave the same customer then at most one of them can be used, and if one of them is used then the cost of the set must be accounted for.

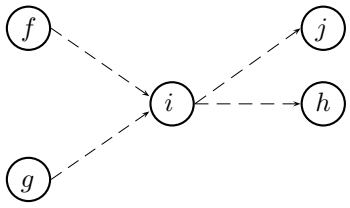


Figure 7.1: Bound on outgoing edges

From this observation we can construct the constraints:

$$\sum_{v \in K} \sum_{j \in C : (i,j) \in \mathbf{A_r}} x_{ij}^v \leq y_r \qquad \forall r \in R, i \in C_r \tag{7.12}$$

Where $C_r \subseteq C$ contains the customers of C with an out edge in set r. In this case the integrality of the x variables again imposes the integrality of the y variables and the number of constraints in (7.12) is at most $|C||R|$. Note that constraints (7.12) do not apply to the depot as more than one edge belonging to a group may leave the depot. Therefore the constraints of type (7.12) cannot entirely replace the constraints (7.2). However, by formulating a new set of constraints (see (7.13)) similar to the constraints (7.12) for edges entering every customer then the constraints (7.2) can be replaced by $2|C||R|$ constraints or less.

This means that the constraints (7.2) in the model can be replaced by the constraints:

$$\sum_{v \in K} \sum_{(j,i) \in \mathbf{A_r}} x_{ji}^v \leq y_r \qquad \forall r \in R, i \in C_r \tag{7.13}$$

$$\sum_{v \in K} \sum_{(i,j) \in \mathbf{A_r}} x_{ij}^v \leq y_r \qquad \forall r \in R, i \in C_r \tag{7.14}$$

These tighter constraints will in the following replace constraints (7.2) in the model.

7.4 Solution method

The branch-cut-and-price method has with success been applied to the VRPTW and the best results for finding exact solutions to the problem have been produced using this method (see Jepsen et al. [12], Desaulniers et al. [8] and Baldacci et al. [1]). Since the ESVRPTW is a generalization of the VRPTW the solution methods for the VRPTW may successfully be applied to the ESVRPTW. Therefore we will apply the BCP algorithm to the VRPTW using cuts for the original formulation of the VRPTW presented by Fukasawa et al. in [11] and by Lysgaard et al. [19] for the CVRP. This corresponds to the algorithm developed by Jepsen et al. [12] for the VRPTW. Jepsen et al. also introduced the Subset Row valid inequalities into the master problem formulation. We will later argue that these cuts can with the same benefits be applied to the ESVRPTW.

The model for the ESVRPTW can be decomposed into a master and pricing problem the similar to the Dantzig-Wolfe decomposition of the standard model for the VRPTW where, the pricing problem is to find a elementary shortest path problem with resource constraints.

7.4.1 Master Problem

The master problem is similar to the standard VRPTW decomposition master problem presented by Desrochers et al. [9]. However, the cost of the edge sets are kept in the master problem and these costs will be reflected in the dual variables from the solution of the linearly relaxed master problem.

$$\textbf{Min:} \sum_{p \in P} \sum_{(i,j) \in \mathbf{A}} c_{ij} \alpha_{ijp} \lambda_p + \sum_{r \in R} c_r y_r \tag{7.1}$$

$$s.t. \quad \sum_{p \in P} \sum_{(j,i) \in \mathbf{A_r}} \alpha_{jip} \lambda_p \leq y_r \qquad \forall i \in C, \forall r \in R \tag{7.2}$$

$$\sum_{p \in P} \sum_{(i,j) \in \mathbf{A_r}} \alpha_{ijp} \lambda_p \leq y_r \qquad \forall i \in C, \forall r \in R \tag{7.3}$$

$$\sum_{p \in P} \sum_{(j,i) \in \mathbf{A}} \alpha_{jip} \lambda_p = 1 \qquad \forall i \in C \tag{7.4}$$

$$\lambda_p \in \{0,1\} \qquad \forall\, p \in P \tag{7.5}$$

$$y_r \in \{0,1\} \qquad \forall\ r \in R \tag{7.6}$$

The set P contains routes satisfying the time window constraints and the capacity constraints. When λ_p is one then route $p \in P$ is used and λ_p is zero otherwise. The constant α_{ijp} is one if the edge $(i,j) \in A$ is used by the route p and zero otherwise. Constraints (7.2) and (7.3) corresponds to the constraints (7.13) and (7.14) which ensure that access to the edges used is paid once if an edge is used in one of the selected routes. Constraints (9.2) ensure that every customer is visited exactly once by the set of routes selected. The master problem can be recognized as a set partitioning problem with side constraints. It is important to note that the constraints (7.2) and (7.3) do not change the domain of valid solutions but only affect the value of the solutions.

7.4.2 Sub problem

The linear relaxation of the master problem can be solved through delayed column generation. The pricing problem is the elementary shortest path problem with resource constraints. Let $\phi'_{ir} \in \mathbb{R}_0^-$ be the dual variables of constraints (7.2) and let $\phi_{ir} \in \mathbb{R}_0^-$ be the dual variables of constraints (7.3). Let $\pi_j \in \mathbb{R}$ be the dual variables of constraint (9.2) and let $\pi_0 = 0$, $\phi'_{0r} = 0$ and $\phi_{0r} = 0$. Then, the reduced cost for a route in the pricing problem becomes:

$$\bar{c}_p = \sum_{(i,j)\in\mathbf{A}} c_{ij}\alpha_{ijp} - \sum_{(i,j)\in\mathbf{A}} \pi_j\alpha_{ijp} - \sum_{r\in R}\sum_{(j,i)\in A_r} \phi'_{ir}\alpha_{jip} - \sum_{r\in R}\sum_{(i,j)\in A_r} \phi_{ir}\alpha_{ijp} \tag{7.7}$$

$$= \sum_{(i,j)\in\mathbf{A}} (c_{ij} - \pi_j)\alpha_{ijp} - \sum_{r\in R}\sum_{(i,j)\in A_r} (\phi_{ir} + \phi'_{jr})\alpha_{ijp} \tag{7.8}$$

This can be transformed to the elementary shortest path problem with resource constraints (ESPPRC) where each edge (i,j) has the cost $\bar{c}_{ij} = c_{ij} - \pi_j - \sum_{\{r|(i,j)\in A_r\}}(\phi_{ir} + \phi'_{jr})$. The resource constraints included in the elementary shortest path problem are the demand picked up along the route and the time accumulated along the route. The demand of the customers visited by the route must be less than or equal to the capacity and the customers must be visited within their time window. The ESPPRC pricing problem can be solved by a label-setting bidirectional shortest path algorithm developed by Righini and Salani [21]. The dominace criteria presented by Desaulniers et al. [7] for removing all labels which are not efficient Pareto optimal, given that the resources are additive or the function on them is monotone, can be used here. However, when introducing the SR cuts which will be described later the objective is no longer additive and the function used is not monotone. In [3] Blander Reinhardt and Pisinger cover several different ESPPRCs with objectives containing functions which are not monotone.

Before running the label setting algorithm timewindow reduction described by Desrochers et al. [9] is run to reduce the complexity of the label setting algorithm. However the ESPPRC is shown by Dror in [10] to be NP-Hard in the strong sense and therefore it is desirable to try to find improving paths without solving the ESPPRC. To do this a simple heuristic is implemented and if the heuristic does not find any improving paths the ESPPRC is solved to optimality using the label setting algorithm. The heuristic used is a simple greedy heuristic which always extends the label with the lowest cost.

7.4.3 Cuts

After adding route variables to the master problem it is investigated if cuts can be added to the master problem. If the added cuts are valid inequalities derived from the original formulation (7.2) to (7.10) then the dual can be transferred directly to the cost of the arcs. Such cuts could be capacity inequalities, strengthened capacity inequalities, framed capacity inequalities, strengthened comb inequalities, multi star inequalities and generalized large multi star inequalities. However, if the cuts added are in the form of the paths variables the dual cost can be more complicating to transfer as the dual of the constraints may be activated not only by a single edge but a combination of edges. However, the subset row cuts have with success been introduced into the master problem variables by Jepsen et al. [12]. Jepsen et al. [12] developed a method of handling the reduced cost of a route for the ESPPRC where the objective contains a function which is not strictly in- or decreasing as a result of the reduced cost achieved from the Subset Row cuts.

7.4.3.1 Valid Inequalities in the original form

Many valid inequalities have been developed for the VRPTW. These valid inequalities will also be applicable for ESVRPTW as the ESVRPTW does not change the set of feasible solutions but only changes the objective function. Valid inequalities for the VRPTW are described in [11], [17] and [19]. The valid inequalities in the original form applied are, as mentioned previously, the capacity inequality, the strengthened capacity inequality, the framed capacity inequality, the strengthened comb inequality, the multi star inequality and the generalized large multi star inequality. These have all been developed for the capacitated vehicle routing problem CVRP but also apply to the VRPTW. However, since they are developed for the CVRP they do not include the time window restrictions to possibly tighten inequalities. The separation algorithm used is that described by Lysgaard et al. [19] and accessible in the framework developed by Lysgaard [18].

7.4.3.2 Valid Inequalities in the master problem form

In [12], Jepsen et al. developed the Subset-Row (SR) inequalities to generate cuts in the set partitioning formulation of the master problem. The SR inequalities are inspired by the clique and odd hole inequalities for the set-packing problem.

The inequalities are not based on the edges directly but on the route variables and formulated as follows:

$$\text{(Subset-Row:)} \quad \sum_{p \in P} \left\lfloor \frac{1}{k} \sum_{i \in S} \alpha_{ip} \right\rfloor \lambda_p \leq \left\lfloor \frac{|S|}{k} \right\rfloor \tag{7.9}$$

Where S is a subset of the constraints (9.2) and $0 < k \leq |S|$ and

$$\alpha_{ip} = \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp}$$

where $\delta^+(i)$ is the edges leaving the vertex i. Clearly, if λ_p has a binary value satisfying the customer constraint (9.2) then the SR inequalities (7.9) will also be satisfied. However, when solving the linearly relaxed master problem λ_p are relaxed to continuous variables between zero and one then there might be solutions where the inequalities (7.9) are not satisfied. These inequalities are limited to the set of (9.2) constraints and can therefore easily be introduced in the ESVRPTW. Moreover the effect from introducing the SR cuts into the ESVRPTW should be the same as in the VRPTW as the set of feasible solutions do not differ between the ESVRPTW and the VRPTW.

The problem with these inequalities is that the dual of each inequality can not be mapped directly to the cost of the individual edges. Using the notation from Jepsen et al. [12] we let the dual variable of a SR inequality be σ we can then formulate the dual cost of a route p as $\hat{c}_p = \bar{c}_p + \sigma \left\lfloor \left(\sum_{i \in S} \sum_{(i,j) \in \delta^+(i)} \alpha_{ijp} \right) / k \right\rfloor$. Note that the dual variable σ is not activated before at least k vertices in the set S has been visited. Therefore to introduce the reduced cost of these constraints into the pricing problem the ESPPRC needs to be modified. As mentioned in Section 9.2.2, the ESPPRC is solved with a label-setting algorithm using dominance for the time, load and cost criteria. Let L be a label at a node v so that each label L at v represents a path from the depot to v. The usual dominance criteria which holds for additive costs is that a label is dominated if there exists another label at the same vertex where all criteria are less than or equal to the dominated labels criteria values. However, this does not hold for the reduced cost introduced by the SR inequalities. One label may be better than the other even if the labels have the same or worse cost. We work with the cuts not allowing two or more routes to visit two vertices from a set of three customers, $k = 2$ and $|S| = 3$.

To solve this problem Jepsen et al. [12] modified the domination rule for the cost criteria in the label setting algorithm used for solving the ESPPRC. The modification consists of subtracting the dual variable σ_q from a label L_i. Where the cost of cut q is included in label L_i and not included in dominated label L_j so that the dominance rule for the cost of two labels at the same

vertex becomes:

$$\hat{c}(L_i) - \sum_{q \in Q} \sigma_q \leq \hat{c}(L_j)$$

Where Q represents a subset of the SR cuts for which the route represented by L_i has visited two vertices in the SR cut and the route represented by L_j has not visited two vertices in the SR cut and where $\sigma_q < 0$. The dominance rules for the cost, capacity and time constraints are the standard dominance rules given by Desaulniers et al. [7] are used as the functions are nondecreasing. For further details see Jepsen et al. [12].

7.4.4 Branch-and-cut-and-price

The branch-cut-and-price algorithm is commonly used for solving integer problems to optimality. Below we describe the algorithm with some of the conditions selected in the method used here included. Note that the algorithm starts with the root node.

The branch-cut-and-price algorithm:

Step 1: Choose an unprocessed node. If several unprocessed nodes exists the node with the lowest lower bound is selected. If the lower bound of a node is above the upper bound then the node will be removed from the unprocessed node list.

Step 2: Solve the LP relaxed master problem. Update the lower bound.

Step 3: Search for routes with negative reduced cost using heuristic methods. If any found add columns to the master problem and go to step 2. The heuristic used is a simple greedy heuristic.

Step 4: Solve the pricing problem to optimality. If routes with negative reduced cost are found then add them to the master problem and go to step 2. If no routes with negative reduced cost are found then the lower bound of the node is updated. If the updated lower bound is above the global upper bound then the node is deleted. Goto step 1.

Step 5: If any violated cuts are found then add them to the master problem and go to step 2

Step 6: Mark the node as processed. If the solution to the LP relaxed master problem is integer then update the upper bound. If the solution is fractional then branch and add the children to the list of unprocessed nodes. Go to step 1.

Note that in the pricing problem the timewindow reduction described by Desrochers et al. [9] is applied. This will also eliminate infeasible arcs. In Step

The branching used is described in more detail in the next subsection.

7.4.5 Branching

For VRPTW the branching is most commonly done on the arc variables remaining after preprocessing. We have chosen to do branching on the group variables as well. Branching on the group variables can reduce the depth of the search tree as the edges to branch on are restricted. Moreover the number of group variables is comparably small. In addition the branching of Fukasawa et al. [11] is implemented which branches on the number of vehicles servicing a set of nodes/customers. This branching can be formulated by letting $S \subset C$ be a subset of the set of arcs remaining after preprocessing and one branch is $\sum_{v \in K} \sum_{(i,j) \in \delta^+(S)} (x_{ij}^v + x_{ji}^v) = 2$ where $\delta^+(S)$ is the edges leaving the set S and the other branch where at least two vehicles services the set S is represented by constraints $\sum_{v \in K} \sum_{(i,j) \in \delta^+(S)} (x_{ij}^v + x_{ji}^v) \geq 4$. To separate candidate sets the Lysgaard cut library [18] is used. From preliminary tests it was clear that branching on the group variables tended to improve the solution time for the problem significantly.

7.5 Closely related formulation and problems

Clearly, there are different versions of the problem where the edges are part of a set. It has been assumed that the edges only belong to one set, however there could be cases where the edges belong to more than one set. If an edge can belong to one set there can be different ways to add the charge. The simple choice would be that the cost must be paid for all the sets an edge belongs to. This problem can be formulated with the constraints (7.13) and (7.14) with the only difference that an edge may be included in more than one constraint for each node. Other alternatives are discussed in this section.

7.5.1 Edges belonging to multiple sets

Another variant could be that for an edge belonging to several sets, the access cost only needs to be paid for one of the sets to which the edge belongs. This can be formulated as:

$$x_{ij} - \sum_{\{r|r\in R\wedge(i,j)\in A_r\}} y_r \leq 0 \;\; \forall (i,j) \in \mathbf{A} \tag{7.1}$$

This constraint is very similar to constraints (7.2); however, in this case the integrality of the x_{ij} variables does not necessarily imply integrality of the y_r variables. Note, that when replacing constraints (7.2) and (7.3) with (7.1) each edge (i, j) in the ESPPRC sub problem will have cost $c_{ij} - \pi_j - \rho_{ij}$ where ρ is the dual variable for the constraints (7.1). This will not add any complications to the ESPPRC algorithm as the cost of a path remains additive and the non additive cost introduced by the SR cuts are handled as in the VRPTW.

7.5.2 Accessing a set of reduced prices

In some cases one may access edges belonging to a set without paying for accessing the set but by paying a more expensive price for using each edge. For instance, instead of buying company access to all freeways in a country, one may be allowed to pay with cash at the barrier to each road. These cash prices are expensive but may be attractive if there is a very limited usage of the edges in the set. This extension is easily handled in our model by duplicating each edge, where one edge belongs to an edge set, and the other edge correspond to cash payment.

$$x_{ij}^{z_r} - \sum_{\{r|r\in R\wedge(i,j)\in A_r\}} y_r \leq 0 \;\; \forall (i,j) \in \mathbf{A} \tag{7.2}$$

where $x_{ij}^{z_r}$ is the edge between i and j which becomes accessible when paying the price for accessing the set r.

In this case each edge $x_{ij}^{z_r}$ in the ESPPRC will have the cost $c_{ij}^{z_r} - \pi_j - \zeta_{ij}$ where ζ_{ij} is the dual variable of the respective constraint of type (7.2) and the cost of the edges not in sets will simply have the cost $c_{ij} - \pi_j$.

7.6 Test data

Following the tradition of the VRP, the test data are based on the Solomon instances [22] making it possible to relate our results to the existing literature. We have generated test instances based on the RC201 to RC204 and C101 to C109 instances by assigning subsets of edges to disjoint groups, and associating a fixed cost with each group. For the RC201 to RC204 Solomon instances different categories of test instances have been constructed. The instances can be grouped into two categories:

1. **random sets**: In these instances, the edges in each set are randomly selected. These instances should reflect a toll on accessing bridges, tunnels or ferries. These facilities are randomly scattered in the plane, but frequently a set of facilities is run by the same operator.

2. **spanning tree sets**: In these instances the selected edges form cheap spanning graphs of a randomly selected subset of vertices. Each subset of vertices consists of half of the total number of vertices. This case should reflect payment of toll on motorways. Motorways usually form a spanning network covering the main cities in a country.

In all test cases, each edge is assigned to at most one edge set.

For each Solomon instance, test instances containing 3, 5 and 8 edge sets were generated, each having an associated cost for accessing the set.

For the **random edge sets** instances, 50% of the edges are assigned to groups with an additional cost. For each combination of Solomon instance and number of groups, two test instances were generated: one case with the costs of an edge set group calculated as $\beta = 5\%$ of the average cost of the edges in the group multiplied by the number of vertices in the set.

In the case of **spanning tree sets**, the cost of a given set is chosen as the most expensive edge in the graph minus the average value of the edges in the given set. This implies that sets containing cheaper spanning trees (i.e. fast transportation times) are more costly than the sets containing more expensive spanning trees.

For the Solomon instances RC201 to RC204, test cases were generated with 15, 20, 30 and 40 customers. For the instances C101 to C109, test cases were generated containing 50 customers using **random edge sets**. We only consider test cases up to 40 customers for the RC201 to RC204 instances, since many instances with 40 customers were not solvable within the given time limit. Instead we have run larger instances with 50 customers for the C101 to C109 instances, as these are know to be easier from the VRPTW literature.

7.7 Results

The program has been implemented in C++ using the COIN bcp library and CLP as the linear programming solver. The test have been run on a Linux machine with a 64 bit Intel Xeon 2.67 GHz CPU. The edge set constraints have been implemented in the framework for vehicle routing problems with time windows by Jepsen et al. [12] provided to us by the authors. On the RC201 - RC204 tests with 20 customers we have tested the effect of running branch-cut-and-price with the cuts implemented in [18] only, the SR cuts [12] only, both the cuts from [18] and SR cuts, and without any cuts. In Table 7.1 the solution times for the four algorithms and for CPLEX are shown. In about half of the instances, CPLEX is not able to solve the routing problem within the time limit. For all four branch-and-price and branch-cut-and-price algorithms the solution was found within 500 seconds. We have ranked the results of the four branch-cut-and-price algorithms by solution times. For the optimal solution the objective value if optimal solution is found is given in the column "objective" and the number of groups payed access to in the optimal solution is given in the column "groups".

In Table 7.1 the rank of a solution is stated in parenthesis after the solution time. The average of the ranks is calculated for each solution algorithm and shown in the last line of Table 7.1.

It is seen that CPLEX is the fastest for 7 instances, while the branch-cut-and-price algorithm using both VRPTW cuts implemented in [18] and SR cuts is the fastest for 11 instances. The branch-cut-and-price algorithms using only one of the cut families are only fastest for 3 instances. The ranking average clearly shows that the branch-cut-and-price algorithm using both Lysgaard and SR cuts has the best average rank. Therefore the branch-cut-and-price algorithm used for the tests in Tables 7.2, 7.3, 7.4 and 7.5 includes the cuts implemented in [18] and SR cuts.

Tables 7.2, 7.3 and 7.4 consider test instances with **random sets**. In Table 7.2 and Table 7.3 it is seen that the branch-cut-and-price with both Lysgaard and SR cuts often has a significantly reduced running time. However, for all of the instances based on RC201 CPLEX finds the solutions within seconds and always much faster than the branch-cut-and-price algorithm. In Table 7.2 there are two instances with 30 customers which cannot be solved within 7500 seconds using the branch-cut-and-price algorithm. However, more than half of the instances cannot be solved by CPLEX within the time limit of 7500 seconds. For the RC201 to RC204 instances with 40 customers shown

instance		opt solution		solution times				
test	groups	objective	groups	CPLEX	bcp C+SR	Bp	bcp SR	bcp C
rc201	3	4239	3	*0.06	1.22 (3)	1.16(1)	1.22(3)	1.20(2)
rc201	5	4539	4	*0.06	1.09(1)	1.23(4)	1.18(3)	1.16(2)
rc201	8	4878	5	*0.09	1.62(2)	1.52(1)	1.66(3)	1.71(4)
rc202	3	3723	2	473.82	*13.55(1)	14.86(2)	16.85(3)	21.76(4)
rc202	5	4120	3	200.19	10.26(2)	*9.40(1)	11.71(4)	11.51(3)
rc202	8	4166	3	25.54	*9.96(1)	10.13(2)	17.06(4)	11.75(3)
rc203	3	3371	1	-	*29.07(1)	29.19(2)	30.30(3)	54.01(4)
rc203	5	3635	2	-	*44.34(1)	44.66(2)	47.71(3)	58.24(4)
rc203	8	3545	2	-	*75.54(1)	81.25(2)	88.53(3)	100.63(4)
rc204	3	3148	1	-	*130.83(1)	147.04(3)	183.59(4)	136.88(2)
rc204	5	3414	2	-	*123.00(1)	205.80(3)	153.72(2)	308.24(4)
rc204	8	3215	1	-	100.40(3)	*43.06(1)	119.16(4)	54.43(2)
Average Rank					1.50	2.0	3.25	3.17

Table 7.1: RC201-RC204 instances with 20 customers, **random sets**. If the algorithm has not terminated within 7500 seconds it is indicated with "-". The best running time for each instance is marked with a "*". C indicates that the cuts implemented in Lysgaard are used, SR indicates that SR-cuts are used, while C+SR indicates that both families of cuts are used.

instance		opt solution		solution times	
test	groups	objective	groups	CPLEX	Bcp L+SR
rc201	3	7100	3	*1.08	9.30
rc201	5	7173	2	*0.14	13.75
rc201	8	7685	4	*0.41	28.31
rc202	3	5660	2	-	*52.90
rc202	5	5886	2	-	*206.10
rc202	8	5915	3	2897.36	*128.22
rc203	3	5144	1	-	*104.29
rc203	5	5429	1	-	*509.70
rc203	8	5821	2	-	*870.11
rc204	3	4847	1	-	*262.21
rc204	5	-	-	-	-
rc204	8	-	-	-	-

Table 7.2: RC201-RC204 instances with 30 customers, **random sets**. If the algorithm has not terminated within 7500 seconds it is indicated with "-". The best running time is marked with a "*".

instance		opt solution		solution times	
test	groups	objective	groups	CPLEX	Bcp L+SR
rc201	3	8660	2	*0.47	7.73
rc201	5	9431	3	*1.14	62.16
rc201	8	10064	3	*4.85	152.61
rc202	3	7805	1	-	*226.13
rc202	5	8518	2	-	*737.95
rc202	8	8755	2	-	*4316.18
rc203	3	7098	1	-	*430.43
rc203	5	7120	1	-	*1647.74
rc203	8	-	-	-	-
rc204	3	-	-	-	-
rc204	5	-	-	-	-
rc204	8	-	-	-	-

Table 7.3: RC201-RC204 instances with 40 customers, **random sets**. If the algorithm has not terminated within 7500 seconds it is indicated with "-". The best running time is marked with a "*".

in Table 7.3 only the RC201 instances were solved by the branch and bound algorithm within the time limit, and 9 instances were not solved by the branch-cut-and-price algorithm within the time limit.

Table 7.2 and Table 7.3 show the running time for instances generated from RC201 to RC204 with respectively 30 customers and 40 customers. The running times in Table 7.2 show that the branch-cut-and-price for most of the instances runs much faster than CPLEX. The same is true for the running times in Table 7.3 when only considering instances where at least one of the algorithms terminated within the time limit.

Notice, that most of the instances not solved by CPLEX within the time limit were solved by branch-cut-and-price. Half of the instances were solved in less than 600 seconds (10 minutes).

For the C101-C109 instances with 50 customers shown in Table 7.4 the results are more mixed.

CPLEX is the fastest for easy instances, while the branch-cut-and-price algorithm has some computational overhead which only pays off for the difficult instances. The branch-cut-and-price algorithm solves significantly more instances within the time limit, although there are two instances solved by CPLEX which cannot be solved by branch-cut-and-price within the time limit.

Instance		opt solution		Solution Times	
test	groups	objective	groups	CPLEX	Bcp L+SR
c101	3	5683	3	*0.99	37.61
c101	5	6749	2	*1.04	574.34
c101	8	7125	3	*2.27	1037.11
c102	3	5304	2	-	*322.95
c102	5	5741	2	-	*1210.31
c102	8	6201	1	-	*6932.81
c103	3	4801	1	-	*2158.50
c103	5	4979	1	-	*5999.15
c103	8	5081	0	-	*854.73
c104	3	-	-	-	-
c104	5	-	-	-	-
c104	8	-	-	-	-
c105	3	5421	2	*36.90	394.63
c105	5	5838	2	*705.29	1686.57
c105	8	6083	1	*852.82	5477.76
c106	3	5600	2	*2.05	147.36
c106	5	6428	2	*5.82	2652.120
c106	8	6896	2	*4.53	1340.30
c107	3	5204	2	*364.35	439.94
c107	5	5618	1	*2250.54	2945.98
c107	8	5700	1	*220.37	1164.84
c108	3	5076	1	-	*1260.62
c108	5	5325	-	-	*1124.48
c108	8	5628	1	-	*2362.47
c109	3	-	-	-	-
c109	5	-	-	-	-
c109	8	5022	0	-	*2730.11

Table 7.4: C101-C107 instances with 50 customers, **random sets**. If the algorithm has not terminated within 7500 seconds it is indicated with "-". The best running time is marked with a "*".

Table 7.5 contains the test results for instances RC201 to RC204 with **spanning tree sets**. For the RC201 instances, CPLEX solves the instances within a second and considerably faster than the branch-cut-and-price algorithm. For the RC201 to RC204 instances the branch-cut-and-price algorithm solves the problem faster than CPLEX. The last instance has not been solved within the time limit by any of the algorithms. In general the branch-cut-and-price algorithm solves considerably more problems to optimality than CPLEX within the given time limit.

7.8 Conclusion

The vehicle routing problem with time windows and fixed costs for accessing an edge set (ESVRPTW) has been presented in this paper. To the best of our knowledge, it is the first time this type of problem has been investigated. A mathematical model has been presented for the ESVRPTW. We have applied the branch-cut-and-price method to the problem and shown that including the SR cuts and the cuts implemented in Lysgaard [18] for the VRPTW and CVRP improves the solution times for this problem. Many related routing problems may with advantage be implemented this way using the extensive research available for the CVRP and the VRPTW.

Acknowledgments

The authors wish to thank Richard Martin Lusby for valuable comments.

Bibliography

[1] R. Baldacci, A. Mingozzi, and R. Roberti. Solving the vehicle routing problem with time windows using new state space relaxation and pricing strategies. Submitted.

instance			opt solution		solution times	
test	customers	groups	objective	groups	CPLEX	Bcp SR+L
rc201	15	3	2618	1	*0.05	0.75
rc201	15	5	3153	3	*0.04	1.98
rc201	15	8	3474	4	*0.06	1.92
rc202	15	3	2478	2	176.17	*5.03
rc202	15	5	2773	2	33.74	*9.13
rc202	15	8	3154	4	93.06	*40.43
rc203	15	3	2400	0	556.14	*5.37
rc203	15	5	2665	1	268.44	*16.88
rc203	15	8	2982	2	4342.91	*151.65
rc204	15	3	2240	0	-	*6.21
rc204	15	5	2526	1	6698.49	*36.80
rc204	15	8	2859	2	2438.21	*402.83
rc201	20	3	3733	2	*0.02	0.26
rc201	20	5	4302	3	*0.03	1.34
rc201	20	8	4931	3	*0.05	2.42
rc202	20	3	3464	1	107.58	*3.88
rc202	20	5	3862	2	56.45	*1.25
rc202	20	8	4635	4	1276.84	*163.60
rc203	20	3	3042	0	-	*7.09
rc203	20	5	3366	1	-	*136.39
rc203	20	8	4120	3	-	*2986.15
rc204	20	3	2845	0	-	*6.82
rc204	20	5	3301	1	-	*1160.77
rc204	20	8	3790	2	-	*6469.83
rc201	30	3	5599	1	*0.11	4.44
rc201	30	5	6204	2	*0.34	12.57
rc201	30	8	6998	4	*0.24	37.11
rc202	30	3	4832	1	-	*21.95
rc202	30	5	5478	2	-	*99.57
rc202	30	8	6431	5	-	*1482.03
rc203	30	3	4418	1	-	*14.01
rc203	30	5	5191	2	-	*374.77
rc203	30	8	5877	4	-	*4458.23
rc204	30	3	4292	0	-	*24.82
rc204	30	5	4953	2	-	*5606.51
rc204	30	8	-	-	-	-

Table 7.5: RC201-RC204 instances, **spanning tree sets**. If the algorithm has not terminated within 7500 seconds it is indicated with "-". The best running time is marked with a "*".

[2] J.-M. Belenguer, E. Benavent, C. Prins, C. Prdhon, and R. W. Calvo. A branch-and-cut method for the capacitated location-routing problem. *Computers & Operations Reserach*, 38:931–941, 2011.

[3] L. Blander Reinhardt and D. Pisinger. Multi-objective and multi-constraint non-additive shortest path problems. *Computers and Operations Research*, 38:605–616, 2011.

[4] N. Christofides, A. Mingozzi, and P. Toth. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11:145–164, 1981.

[5] W. Cook and J. L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical report, Rice University, 1999.

[6] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

[7] G. Desaulniers, J. Desrosiers, I. Ioachim, M. M. Solomon, F. Soumis, and D. Villeneuve. *A unified framework for deterministic time constrained vehicle routing and crew scheduling problems*, pages 57–93. Springer, 1998.

[8] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle reouting problem with time windows. *Transportation Science*, 42:387–404, 2008.

[9] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.

[10] M. Dror. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42:977–978, 1994.

[11] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragao, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Programming*, 106:491–511, 2006.

[12] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56:497–511, 2008.

[13] B. Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35:2307–2330, 2008.

[14] B. Kallehauge, J. Larsen, and O. B. G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33:1464–1487, 2006.

[15] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.

[16] A. W. J. Kolen, A. H. G. Rinnooy Kan, and H. W. J. M. Trienekens. Vehicle routing with time windows. *Operations Research*, 35:266–273, 1987.

[17] A. N. Letchford, R. W. Eglese, and J. Lysgaard. Multistarts, partial multistars and the capacitated vehicle routing problem. *Math. Programming*, 94:21–40, 2002.

[18] J. Lysgaard. Cvrpsep: A package of separation routines for the capacitated vehicle routing problem, 2003.

[19] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming Serie A*, 100:423–445, 2004.

[20] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177:649–672, 2007.

[21] G. Righini and M. Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3:255–273, 2006.

[22] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

Chapter 8

Bounding component sizes of two-connected Steiner networks

ELSEVIER

ScienceDirect

Information Processing Letters 104 (2007) 159–163

Information Processing Letters

www.elsevier.com/locate/ipl

Bounding component sizes of two-connected Steiner networks

K. Hvam, L. Reinhardt, P. Winter *, M. Zachariasen

Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark

Received 11 October 2006; received in revised form 30 May 2007; accepted 11 June 2007

Available online 20 June 2007

Communicated by S.E. Hambrusch

Abstract

We consider the problem of constructing a shortest Euclidean 2-connected Steiner network in the plane (SMN) for a set of n terminals. This problem has natural applications in the design of survivable communication networks.

In [P. Winter, M. Zachariasen, Two-connected Steiner networks: Structural properties, OR Letters 33 (2005) 395–402] we proved that all cycles in SMNs with Steiner points must have pairs of consecutive terminals of degree 2. We use this result and the notion of reduced block-bridge trees suggested by Luebke [E.L. Luebke, k-connected Steiner network problems, PhD thesis, University of North Carolina, USA, 2002] to show that no full Steiner tree in a SMN spans more than $\lfloor n/3 \rfloor + 1$ terminals.

Keywords: Computational geometry; Interconnection networks; 2-connected Steiner networks

1. Introduction

The Euclidean Steiner tree problem asks for a shortest possible network spanning a set Z of n terminals in the plane. The solution is a tree, referred to as a *Steiner minimal tree* (SMT). Apart from the terminals, SMTs may contain additional, so-called *Steiner points*, where exactly three edges meet at 120° angles. SMTs are unions of *full Steiner trees* spanning subsets of terminals all having degree 1.

When the objective is to design low cost survivable networks, the problem of constructing *Euclidean 2-connected Steiner minimum networks* in the plane (SMNs) arises. Since a 2-edge-connected minimum-length network necessarily is 2-vertex-connected when the distance function is a metric [1,5], we use the shorthand 2-connected in the following.

SMNs have been studied by Hsu and Hu [2], Luebke and Provan [4], Luebke [3] and Winter and Zachariasen [8]. Luebke and Provan [4] proved that the Euclidean 2-connected SMN problem is NP-hard and gave a number of structural properties of SMNs. Luebke [3] introduced the notion of (reduced) block-bridge trees that will play an essential role in this paper. Winter and Zachariasen [8] proved that all cycles in SMNs with Steiner points must have pairs of consecutive terminals of degree 2.

The paper is organized as follows. In Section 2 we formally define the problem and outline some basic properties of SMNs. In Section 3 we discuss reduced block-bridge trees. In Section 4 we show how the properties of the reduced block-bridge trees for SMNs can

* Corresponding author.
E-mail addresses: hvam@diku.dk (K. Hvam), line@diku.dk (L. Reinhardt), pawel@diku.dk (P. Winter), martinz@diku.dk (M. Zachariasen).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2007.06.009

be used to bound the size of full Steiner trees. Concluding remarks are given in Section 5.

2. Basic properties

Let Z be a set of n terminals in the Euclidean plane. The *2-connected Euclidean Steiner network problem* is to find a minimum length 2-connected network $N(Z)$ spanning Z and possibly additional, so called *Steiner points*. If Z is obvious from the context, we denote $N(Z)$ by N.

Hsu and Hu [2], Luebke and Provan [4] and Luebke [3] proved several properties of SMNs. In particular, all Steiner points are incident with three edges meeting at 120° angles (as for the Euclidean Steiner tree problem in the plane). Furthermore, no cycle consists entirely of Steiner points. As a consequence, SMNs are unions of *full Steiner trees (FSTs)*, in which all terminals are leaves and all Steiner points are interior vertices (Fig. 1). This important result implies that `GeoSteiner`, the 2-phase exact algorithm for the determination of SMTs in the plane [7] can be adapted to find SMNs. `GeoSteiner` in its first phase generates a superset of FSTs of a SMT. Powerful geometric tests based on non-trivial properties of SMTs permit to keep this superset very small. In the second phase, `GeoSteiner` uses elaborate branch-and-cut approach to identify FSTs of the superset whose concatenation yields a SMT. Problem instances with up to 10000 terminals can be solved within reasonable amount of time.

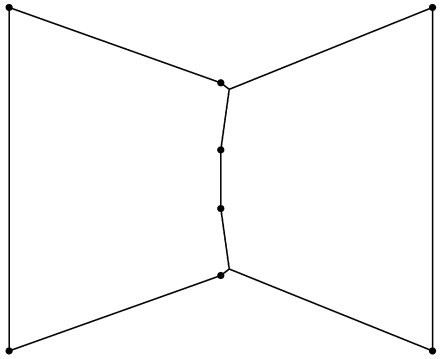


Fig. 1. SMN with 2 non-trivial FSTs of 3 terminals each.

The same approach can be used to find SMNs if the connectivity constraints in the integer programming model for the concatenation phase are appropriately modified. While this modification is very straightforward, the real problem occurs in the first phase where FSTs are generated. Some of the very powerful geometric test that work for the SMTs, do not apply in the 2-connected case.

In [8], we introduced the notion of a *chord-path*. Let G denote an undirected graph and let C be a cycle in G. A chord-path between two distinct vertices u and v on C is a path between u and v in G that shares no interior vertices with C. We proved in [8] the following structural results.

Theorem 1. *Any chord-path in a SMN must have a pair of consecutive terminals of degree* 2 *in its interior.*

Corollary 1. *Suppose that a SMN has vertices of degree* 3. *Any of its cycles has two pairs of consecutive terminals of degree* 2 *separated by vertices of degree* 3.

3. Reduced block-bridge trees

Let H be an undirected connected graph. The *block-bridge tree* H_B of H is a tree obtained by contracting each block (i.e., each 2-connected component) to a vertex (Fig. 2a and 2b). The *reduced block-bridge tree* H_R of a connected graph H, is obtained from the block-bridge tree H_B of H by replacing each simple path (with interior vertices of degree two) by an edge (Fig. 2c). Luebke [3] introduced reduced block-bridge trees and gave a proof of Theorem 2 in her PhD thesis; here we present a modified and shorter version of the proof.

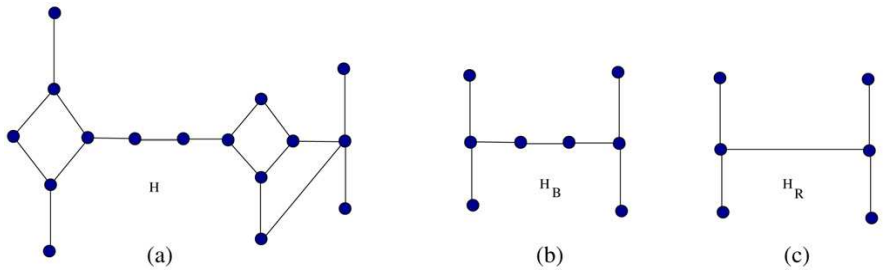


Fig. 2. Reduced block-bridge tree construction.

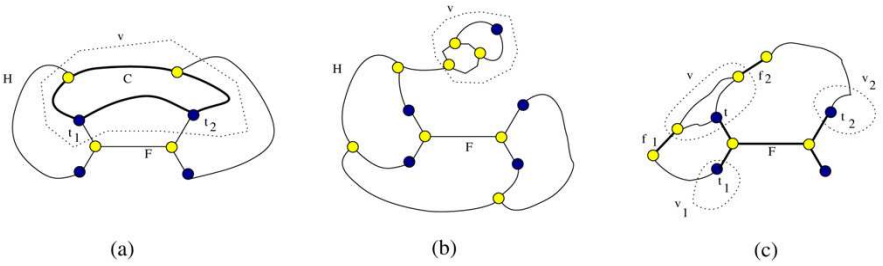


Fig. 3. Impossible contractions.

Given a SMN N of Z and one of its FSTs F, we let $N \setminus F$ denote the graph obtained by deleting from N all interior vertices and all edges of F.

Lemma 1. *Let F be an FST in an SMN N of Z. $H = N \setminus F$ is connected.*

Proof. Suppose that H is disconnected. Let u and v be two vertices of H, each in a different component of H. Since N is 2-connected, it contains two disjoint paths P and P' between u and v. Let C denote the cycle created by P and P'. Both P and P' must each go through at least one vertex of F. Since F is a tree with no terminals in its interior, C has a terminal-free chord-path. This contradicts Theorem 1. □

In the remainder of this paper, we let H_B denote the block-bridge tree of $H = N \setminus F$ and H_R the reduced block bridge tree of H.

Consider an edge e of an arbitrary tree T. When e is removed, T breaks down into 2 subtrees, T_L^e and T_R^e. Let S_L^e and S_R^e denote the leaves in T_L^e and T_R^e, respectively. The pair of sets $S^e = (S_L^e, S_R^e)$ is referred to as a *split* of T by the edge e. It is clear that splits generated by different edges of the same tree are different. It is also well-known that two trees with the same set of leaves (and with no vertices of degree 2) are isomorphic if and only if their edges generate the same splits [6, Theorem 3.1.4].

Lemma 2. *Let F be an FST in a SMN N. There is a one-to-one correspondence between the terminals of F and the leaves of H_R, such that each terminal in F is contracted to a distinct leaf in H_R.*

Proof. First we prove the result for the block-bridge tree H_B and then we extend the result to the reduced block-bridge tree H_R.

Suppose that two distinct terminals t_1 and t_2 of F are contracted to the same vertex v of H_B. Terminals t_1 and t_2 are therefore on a cycle C in H (Fig. 3a). The path connecting t_1 and t_2 in F is a terminal-free chord-path of C, contradicting Theorem 1. Thus at most one terminal of F is contracted to a vertex in H_B.

Suppose that v is a leaf in H_B, and no terminal of F is contracted to it (Fig. 3b). This is only possible if there is a bridge in N, contradicting the assumption that N is 2-connected.

Suppose that v is a non-leaf of H_B and that a terminal t of F is contracted to v. H_B is a tree. There are therefore two paths in H_B, both starting at v but otherwise disjoint, one through edge f_1 and the other through edge f_2, ending in leaves v_1 and v_2 (Fig. 3c). Consequently, there must be terminals t_1 and t_2 in H contracted to v_1 and v_2 respectively. Since the part of H contracted to v in H_B is 2-connected (with special case where the contracted part consists of t alone), there is a path in H from t_1 to t_2 going through t. Since F is also a tree, there must be a path in F between t_1 and t_2. The cycle composed by these two paths has a terminal-free chord-path, a contradiction.

When going from H_B to H_R, every leaf is preserved. All such vertices contain exactly one terminal from F, finishing the lemma. □

Let N_R denote the union of F and H_R (where the terminals in F have been identified with the corresponding leaves of H_R).

Theorem 2. *Let F be an FST in a SMN N. F and H_R are isomorphic.*

Proof. Consider an arbitrary edge $e \in F$. We will first show that $N_R \setminus e$ has a unique bridge f located in H_R.

Let u and v denote the end-vertices of e. Assume that there are 2 vertex-disjoint paths between u and v in $N_R \setminus e$ (Fig. 4a). Then there exists a cycle in N with e being its terminal-free chord-path, a contradiction.

Assume next that there are 2 edge-disjoint (but not vertex-disjoint) paths R_{uv} and R'_{uv} between u and v in

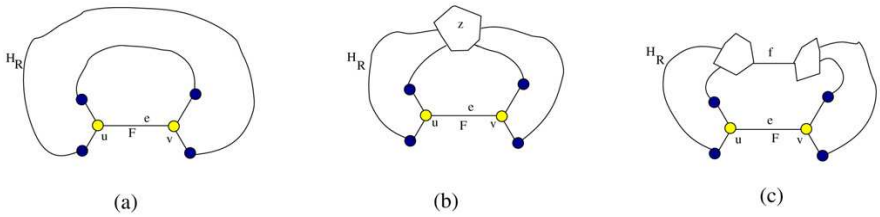


Fig. 4. Isomorphism between F and H_R.

H_R sharing some vertex z (Fig. 4b). The vertex z cannot correspond to a single vertex in H; its degree in H and hence in N would then be at least 4, a contradiction. The vertex z must therefore correspond to a contracted block B of H. Since R_{uv} is a path in H_R, there is a corresponding path P_{uv} in $N \setminus e$. Suppose that P_{uv} enters block B through a vertex u_1 and leaves B through a vertex v_1. If $u_1 = v_1$, then there exists a path P'_{uv} in $N \setminus e$ corresponding to R'_{uv} which is vertex-disjoint with P_{uv} in B (otherwise $u_1 = v_1$ would have degree 4 in $N \setminus e$). If $u_1 \neq v_1$, let C be a cycle in B through the part of P_{uv} between u_1 and v_1. Such a cycle C always exists since B is 2-connected. If P'_{uv} avoids C, then P_{uv} and P'_{uv} are vertex-disjoint in B. Assume that P'_{uv} enters C through a vertex u'_1 and leaves C through a vertex v'_1. Since these vertices must have degrees less than 4, u_1, v_1, u'_1 and v'_1 must be mutually different. No matter in what order they appear on C, there always will exist 2 paths between u and v in $N \setminus e$ that are vertex-disjoint in B (follow P_{uv} and P'_{uv} from u until reaching C at u_1 and u'_1, reach v_1 and v'_1 through disjoint parts of C and continue toward v). The edge-disjoint paths R_{uv} and R'_{uv} can share several vertices. We argued so far that there are paths P_{uv} and P'_{uv} in $N \setminus e$ that are vertex-disjoint in a block corresponding to a shared vertex. Hence, there exist paths P_{uv} and P'_{uv} that are vertex-disjoint in *all* these blocks. Such P_{uv} and P'_{uv} will form a cycle with e as its chord-path, a contradiction.

We have shown so far that $N_R \setminus e$ contains at least one bridge f (Fig. 4c). Since F and N_R are trees sharing only their leaves, f must be located in H_R. We will now show that f is unique. $N_R \setminus \{e, f\}$ is disconnected. As a consequence, the split generated by e in F must be the same as the split generated by f in H_R. Since no pair of splits in a tree (with no vertices of degree 2) is the same [6], it follows that f is unique.

It can be shown in a similar manner that for any $f' \in H_R$, the graph $H_R \setminus f'$ contains a unique bridge e' located in F.

We proved that there is a unique perfect matching between edges of F and edges of H_R, such that for every pair of matched edges $e \in F$ and $f \in H_R$, e is a bridge in $N_R \setminus f$ if and only if f is a bridge in $N_R \setminus e$. Since F and H_R share their leaves, the splits generated by the matched edges are the same. This implies that F and H_R are isomorphic. □

4. Size of full Steiner trees

In this section we use reduced block-bridge trees to obtain an upper bound on the number of terminals in FSTs of a SMN. Such a bound can be used to make the generation phase of the exact algorithm more efficient. It also influences the concatenation phase as the number of FSTs becomes smaller.

Theorem 3. *A SMN N spanning n terminals has no FST spanning more than $\lfloor n/3 \rfloor + 1$ terminals.*

Proof. Let F denote an FST of N. Let m denote the number of terminals in F. Let H, H_R and N_R be as defined in Section 3.

Let $e \in F$. By Theorem 2, there is a unique edge $f \in H_R$ such that e is a bridge in $N_R \setminus f$ and f is a bridge in $N_R \setminus e$. Furthermore, $N_R \setminus \{e, f\}$ consists of two 2-connected components.

Let H_f denote the subgraph of H corresponding to the edge $f \in H_R$. H_f can be viewed as a sequence of vertices and 2-connected components of H. Suppose that H_f has no interior 2-connected components or terminals in this sequence. Hence, f corresponds to an edge in H. The removal of e and f from H_R leaves two 2-connected components. But this implies that the removal of e and f from N also leaves two 2-connected components, contradicting the fact observed by Monma et al. [5] that deleting any pair of edges of an SMN will leave a bridge in one of the resulting components.

Suppose next H_f has an interior 2-connected component and that it contains no terminal. Then such a component can be replaced in H by a single edge. The new graph remains 2-connected and due to the triangle inequality it is shorter, a contradiction.

It follows that H_f must contain at least one terminal in its interior. F has $2m - 3$ edges. Since F and

H_R are isomorphic, H_R also has $2m - 3$ edges. Each of these edges corresponds to a subgraph of H containing at least one terminal. As a consequence, N contains at least $3m - 3$ terminals (m of them belonging to F). Hence, $n \geqslant 3m - 3$ and it follows that $\lfloor n/3 \rfloor + 1$ is an upper bound on the number of terminals in any FST of N. □

5. Concluding remarks

The above result can be used in the 2-phase algorithm in a very straightforward manner. FSTs are generated during the first phase of the algorithm in non-decreasing order of the number of terminals they span. So the generation can be cut off when the number of terminals gets beyond $\lfloor n/3 \rfloor + 1$. This upper bound is normally not as good as the upper bound that can be computed by solving a series of travelling salesman problem instances as described in [8]. But it is readily available and does not require solving any difficult combinatorial optimization problems.

References

[1] G.N. Frederickson, J. Ja'Ja, On the relationship between the biconnectivity augmentation and traveling salesman problem, Theoretical Computer Science 13 (1982) 189–201.
[2] D.F. Hsu, X.-D. Hu, On shortest two-edge connected Steiner networks with Euclidean distance, Networks 32 (1998) 133–140.
[3] E.L. Luebke, k-connected Steiner network problems, PhD thesis, University of North Carolina, USA, 2002.
[4] E.L. Luebke, J.S. Provan, On the structure and complexity of the 2-connected Steiner network problem in the plane, OR Letters 26 (2000) 111–116.
[5] C.L. Monma, B.S. Munson, W.R. Pulleyblank, Minimum-weight two-connected spanning networks, Mathematical Programming 46 (1990) 153–171.
[6] C. Semple, M. Steel, Phylogenetics, Oxford Series in Mathematics and Its Applications, vol. 24, Oxford University Press, 2003.
[7] D.M. Warme, P. Winter, M. Zachariasen, Exact algorithms for plane Steiner tree problems: A computational study, in: D.-Z. Du, J.M. Smith, J.H. Rubinstein (Eds.), Advances in Steiner Trees, Kluwer Academic Publishers, 2000, pp. 81–116.
[8] P. Winter, M. Zachariasen, Two-connected Steiner networks: Structural properties, OR Letters 33 (2005) 395–402.

Chapter 9

Root Balanced Minimum Spanning Graph: Algorithm and Complexity

Line Blander Reinhardt* David Pisinger* Amelia Regan†

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
lbre@man.dtu.dk, pisinger@man.dtu.dk

†Donald Bren School of Information and Computer Sciences,University of California, Irvine,
6210 Donald Bren Hall,Irvine, CA 92697-3425
aregan@uci.edu

Abstract Let $G(E,V)$ be a cycle-free graph where $V = \{0,\ldots,n+1\}$ is the vertex set, and $E = \{(i,j)|i,j \in V\}$ is the set of edges. For each edge (i,j) let c_{ij} be the associated cost. The Minimum Root Balanced Spanning Graph Problem on G is to find a minimum weight subgraph of G which contains a spanning tree on the vertices 0 to n where the number of edges between the vertex sets $\{1,\ldots,n\}$ and $\{n+1\}$ is equal to the number of edges between $\{0\}$ and $\{1,\ldots,n\}$. This problem with capacity and time window constraints appears as a subproblem in a decomposition of the vehicle routing problem with time windows. An algorithm to solve this problem in $O(|E|\alpha(|E|,|V|)+|V|\log|V|)$ time, where α is the inverse Ackermann's function, is presented. Moreover we prove that the shortest spanning arborescence with time windows is NP complete. A prototype solution algorithm for solving the shortest spanning arborescence problem with time windows is presented.

9.1 Introduction

In this paper we will define the root balanced minimum spanning graph problem (RBMSG) and present an algorithm for solving it. We will examine the potential of using RBMSG with capacity constraints and time windows (RBMSGTW) as a subproblem when using decomposition to solve the vehicle routing problem with time windows (VRPTW). Moreover, we will examine the complexity of the subproblem of the VRPTW decomposition.

9.1.1 The Vehicle Routing Problem with Time Windows

For a set of vehicles starting and ending at a single depot, the VRPTW is the problem of selecting the cheapest set of routes so that all customers are serviced within their available time window

while satisfying that the load on the vehicles is less than the capacity. The problem has symmetric edges by definition.

This problem occurs in distribution and also in berth scheduling, airport landing scheduling, home care crew scheduling and many other areas. Being able to solve the VRPTW to optimality could lead to large savings and/or better service for airports, maritime ports, distributors and home care agencies. The wide applicability of the VRPTW has lead to extensive research in the area as can be seen in [1], [4], [10], [11], [13], [14] and [15].

Even though the VRPTW is a very common problem it is hard to solve. To test the quality of VRPTW algorithms, a benchmark set was created by Solomon [18]. To this day the instance R208 in this test suite has yet to be solved to optimality for 100 customers and there are instances for which it still takes computers a while to solve. For a recent overview of solved instances and solution times see Jepsen et al. [11] and Baldacci et al. [1].

The mathematical model for the VRPTW is based on the model presented in [11]. Given the following sets:

C	The set of customers
V	The set vertices representing the customers in C and the depot defined as 0
$\mathbf{A}$	The set of arcs (i, j) in V
K	The set of vehicles

Let the depot be represented by two vertices 0 and $n+1$, where 0 is the start vertex of the depot and $n+1$ is the end vertex representing the depot. The variables are defined as:

x_{ij}^v	Indicator variable indicating if the arc (i, j) is used by vehicle $v \in K$
t_i^v	The time vehicle v visits $i \in V$.

The parameters are defined as:

D	The capacity of the vehicles
d_i	The demand to be delivered to vertex $i \in V$. The demand at depot is zero
a_i	The availability time for customer $i \in C$
b_i	The required completion time for customer $i \in C$
c_{ij}	The cost of using an arc $(i, j) \in A$

Then the VRPTW can be formulated as follows:

$$\textbf{Min:} \sum_{v \in K} \sum_{(i,j) \in \mathbf{A}} c_{ij} x_{ij}^v \tag{9.1}$$

$$s.t. \sum_{v \in K} \sum_{(i,j) \in \mathbf{A}} x_{ij}^v = 1 \qquad \forall i \in C \tag{9.2}$$

$$\sum_{v \in K} \sum_{(j,i) \in \mathbf{A}} x_{ji}^v = 1 \qquad \forall i \in C \tag{9.3}$$

$$\sum_{i \in C} x_{in+1}^v = \sum_{i \in C} x_{0i}^v \qquad \forall v \in K \tag{9.4}$$

$$\sum_{(ij) \in \mathbf{A}} d_i x_{ij}^v \leq D \qquad \forall v \in K \tag{9.5}$$

$$a_i \leq t_i^v \leq b_i \qquad \forall\, i \in V, v \in K \tag{9.6}$$

$$(t_i^v + \theta_{ij}) x_{ij}^v - t_j^v \leq 0 \qquad \forall\ v \in K,\ (i, j) \in \mathbf{A} \tag{9.7}$$

$$x_{ij}^v \in \{0, 1\} \qquad \forall\, (i, j) \in \mathbf{A},\ v \in K \tag{9.8}$$

$$t_i^v \in \mathbb{Z}_0^+ \qquad \forall\, i \in V,\ v \in K \tag{9.9}$$

Where constraints (9.2) and (9.3) ensure that all customers are visited exactly once. Constraints (9.4) ensure that every vehicle used returns to the depot. Constraints (9.5) ensure that the load on the vehicle does not exceed the capacity. Constraints (9.6) formulate the time window constraint on the customers. Constraints (9.7) ensure that the time the vehicle visits a customer is greater than the time the vehicle left the last customer and the travel time, these constraints together with constraint (9.2), (9.3) and (9.4) ensure that the path is connected and starts and ends at the depot $\{0, n+1\}$.

The current most successful exact method for solving the Vehicle Routing Problem with time windows (VRPTW) to optimality is to use Danzig-Wolfe decomposition. Most commonly the VRPTW is decomposed into a set partitioning master problem and a capacitated elementary shortest path problem with time windows (CESPPTW) in the pricing problem. In such a decomposition the subproblem is to find a good capacitated elementary path with time windows (CEPTW). The master problem in that case is to select a set of paths so that each customer is visited exactly once. The paths are generated by the subproblem and introduced into the collection of sets in the master problem when their inclusion can improve the solution of the master problem.

This decomposition method has been very successful in solving VRPTW, but as mentioned earlier the size of solvable problems is still quite limited (For state of the art algorithms see Baldacci et al. [1]).

A problem encountered when using this method is that the elementary shortest path problem is quite hard to solve when there are negative cycles in the graph. Negative weights and cycles can occur on the edges when the weights are adjusted to generate the best fit column.

To circumvent the problem of negative cycles it is suggested by Kallehauge [12] for the VRPTW to have constraints (9.2) in the master problem and a subproblem represented by the remaining constraints. By the decomposition performed here it can be seen that this creates a set partitioning master problem and a subproblem which is the capacitated root balanced minimum spanning graph with time windows and capacity constraints. This decomposition of the VRPTW is lightly described in [12]. In [19], Toth and Vigo implemented the decomposition for the case for the capacitated vehicle routing problem (CVRP). In the CVRP the number of vehicles is fixed and in [19] the capacity restriction was included in the problem but not the time window constraint. The motivation for using the root balanced minimum spanning graph (RBMSG) is that the bounds on the running time of the root balanced minimum spanning graph are not influenced by negative edge weights or cycles. To our knowledge the RBMSG has not been formulated before.

We will start by presenting some simple properties of the RBMSG. In Section 9.3 we will describe the RBMSG. In section 9.4 we will describe the decomposition of the VRP which contains the RBMSG problem as a subproblem using a mathematical model and show results which demonstrate the potential of using the RBMSG problem as a subproblem for the VRP. In section 9.5 we prove that the shortest spanning arborescence with time windows is NP complete by using the NP completeness of the capacitated minimum spanning tree which is proven to be NP complete by Papadimitriou in [16]. Finally in Section 9.6 we conclude and discuss future work.

9.2 A new Danzig-Wolfe decomposition of the VRPTW

As mentioned by Kallehauge in [12] the presented VRPTW model can be Danzig-Wolfe decomposed by keeping constraint (9.2) in the master problem. In this section we will in detail describe this decomposition.

9.2.1 Master Problem

The master problem is similar to the standard VRPTW decomposition master problem presented by Desrochers et al. [5]. The constraints (9.2) are kept in the master problem and the out degrees will be reflected in the dual variables from the solution of the linearly relaxed master problem.

$$\textbf{Min:} \sum_{m \in M} \sum_{(i,j) \in \mathbf{A}} c_{ij} \alpha_{ijm} \lambda_m \tag{9.1}$$

$$s.t. \quad \sum_{m \in M} \sum_{(i,j) \in \mathbf{A}} \alpha_{ijm} \lambda_m = 1 \qquad \forall i \in C \tag{9.2}$$

$$\lambda_m \in \{0,1\} \qquad \forall\, m \in \mathbf{M} \tag{9.3}$$

The set M contains all root balanced minimum arborescence satisfying the time window constraints and the capacity constraints (RBMSGTW). When λ_m is one then the root balanced arborescence with time window and capacity constraints (RBSGTW) $m \in R$ is used otherwise λ_m is zero. The constant α_{ijm} is one if the edge $(i,j) \in A$ is in the RBSGTW m and zero otherwise. Constraints (9.2) ensure that every customer is left exactly once by the set of routes selected. The master problem can be recognized as a set partitioning problem.

Note that by replacing the set M with the set M' containing all the CEPTW one gets the classical Dantzig-Wolfe decomposition of the VRPTW. Moreover in each RBSGTW in M there will be one or more CEPTW and any CEPTW in M' will be contained in at least one RBSGTW in M.

9.2.2 Sub problem

The linear relaxation of the master problem can be solved through delayed column generation. The pricing problem is then the root balanced minimum spanning graph with capacity and time window constraints (RBMSGTW). Let $\pi_i \in \mathbb{R}$ be the dual variables of constraint (9.2) and let $\pi_0 = 0$. Then, the reduced cost for a route in the pricing problem becomes:

$$\bar{c}_m = \sum_{(i,j) \in \mathbf{A}} c_{ij} \alpha_{ijm} - \sum_{(i,j) \in \mathbf{A}} \pi_i \alpha_{ijm} \tag{9.4}$$

$$= \sum_{(i,j) \in \mathbf{A}} (c_{ij} - \pi_i) \alpha_{ijm} \tag{9.5}$$

This can be transformed to the elementary shortest path problem with resource constraints (ESPPRC) where each edge (i,j) has the cost $\bar{c}_{ij} = c_{ij} - \pi_i$. The resource constraints handled by the elementary shortest path problem are the demand picked up along the route and the time accumulated along the route. The demand of the customers visited by the route must be less than the capacity and the customers must be visited within their time window.

9.3 The Root Balanced Minimum Spanning Graph

In the previous sections we have discussed the relevance of the RBMSGTW for the decomposition of the VRPTW. In the following we will discuss the RBMSG as it is the basis of solving the RBMSGTW.

The RBMSG can be formulated as follows:

$$\min \quad c_e x_e \tag{9.1}$$

$$s.t. \qquad \sum_{e \in \{(j,i) | i \in S \wedge j \in V \setminus S\}} x_e \geq 1, \qquad \forall \emptyset \subset S \subset V \tag{9.2}$$

$$\sum_{e \in \{(i,j) | i,j \in C \cup \{0\}\}} x_e \leq |V| - 2 \tag{9.3}$$

$$\sum_{e \in \{i=0 \wedge j \in C\}} x_e - \sum_{e \in \{i=n+1 \wedge j \in C\}} x_e = 0 \tag{9.4}$$

Constraints (9.2) ensure that the selected set of edges form a connected graph. Constraints (9.3) ensure that there are at most $|V| - 2$ edges for which both end points in the set $C \cup \{0\}$ and constraints (9.4) ensure that the number of edges connecting to 0 is the same as the number of edges connecting to vertex $n + 1$. Note that given the root of the RBMSG the direction of the route is implicit and the graph can be considered as undirected for symmetric complete graphs. This formulation has an exponential number of constraints of type (9.2). However considering a directed graph then the problem can be formulated using a polynomial number of constraints. Also note that an undirected graph can be represented by a directed graph but not vice versa. The polynomial directed formulation is:

$$\min \quad c_{ij}x_{ij} \tag{9.5}$$

$$s.t. \qquad \sum_{j \in C \cup \{0\}} x_{ji} \geq 1, \qquad \forall i \in C \tag{9.6}$$

$$\sum_{j \in C \cup \{0\}} x_{0j} \geq 1, \tag{9.7}$$

$$\sum_{i,j \in C \cup \{0\}} x_{ij} \leq |V| - 2 \tag{9.8}$$

$$\sum_{j \in C} x_{0j} - \sum_{j \in C} x_{jn+1} = 0 \tag{9.9}$$

Constraints (9.6) ensure that there is an edge entering each costumer. Constraint (9.7) ensures that there is at least one edge leaving vertex 0. Constraints (9.8) ensure that there are at most $|V| - 2$ edges for which both end points in the set $C \cup \{0\}$ and constraints (9.9) ensures that the number of edges entering $n + 1$ is the same as the number of edges leaving 0. To give a better understanding of the RBMSG we here also give a definition of the RBMSG using natural language and graph notation:

RBMSG: Given a graph $G(E, V)$ a root $0 \in V$ and a sink $n+1 \in V$ and vertices $\{1, \ldots, n\} \in V$ the root balanced minimum spanning graph (RBMSG) problem is to find the minimum weight graph containing a spanning tree of the vertices $V \setminus \{n+1\}$ and the same number of edges from the vertex $n + 1$ to the vertices $\{1, \ldots, n\}$ as from 0 to the vertices $\{1, \ldots, n\}$. There cannot be any edges between 0 and $n + 1$.

When related to the vehicle routing problem the vertex 0 represents the source part of the depot in the vehicle routing problem and the vertex $n + 1$ represents the sink part of the depot. This means that the vehicles leave vertex 0 when starting the deliveries and enter vertex $n + 1$ when they return to the depot from delivering the demands to the customers.

As mentioned in the introduction an RBMSG is a minimum weight graph. Let the set of edges E_r be the edges from the vertex 0 to the vertices $\{1, \ldots, n\}$ in the RBMSG and let the set of edges E_c be the edges between vertices in $\{1, \ldots, n\}$ where the edges with both end points in the set $\{0, 1, \ldots, n\}$ form a spanning tree of the vertices in $\{0, 1, \ldots, n\}$ and the number of edges from $V \setminus \{0, n+1\}$ to vertex $n+1$ is equal to the number of edges from $V \setminus \{0, n+1\}$ to 0. An RBMSG cannot contain any edges between 0 and $n + 1$. Figure 9.1 shows a graph and its corresponding RBMSG.

Before describing the algorithm we will prove some properties needed for the chosen solution approach.

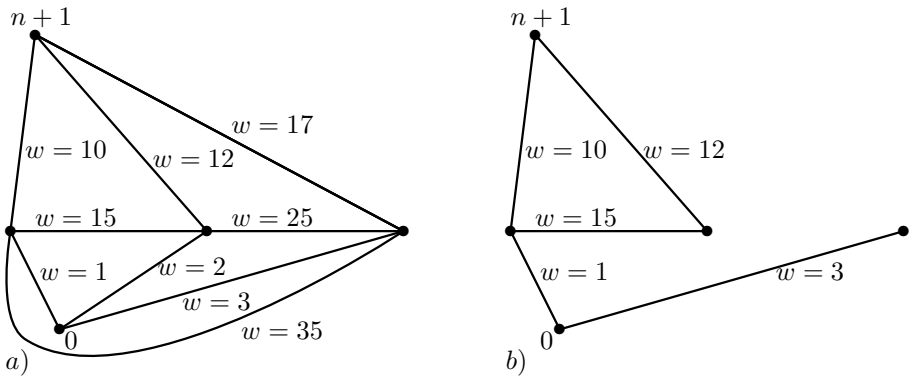


Figure 9.1: a) is a graph G and b) is the RBMSG of G.

Lemma 9.3.1. *Let the RBMSG on a graph $G(V, E)$ have a degree constraint t at the root 0. Then the RBMSG must contain a MST on the vertices $\{0, 1, \ldots, n\}$ with degree constraint t on the vertex 0 and the t smallest edges to vertex $n + 1$ from the vertices in $V \setminus \{0, n + 1\}$.*

Proof. Without loss of generality let the RBMSG contain k edges from the set of vertices $\{1, \ldots, n\}$ to $n+1$ then these edges must clearly be the k shortest as one otherwise could generate a RBMSG of smaller weight by selecting a smaller edge. Note, that there cannot be any edges in an RBMSG from 0 to $n + 1$. Assume that the RBMSG T with a degree constraint t at the root 0 does not contain an MST on the vertices $\{0, 1, \ldots, n\}$ with degree constraint t on the root. Since T is an RBMSG it would contain the t shortest edges from the set of vertices $\{1, \ldots, n\}$ to $n + 1$. Let T' be an RBMSG containing an MST with a degree constraint t on the root. again T' would contain the t shortest edges from the set of vertices $\{1, \ldots, n\}$ to $n+1$. Then by the definition of MST T' would be smaller than T, contradicting that T is an RBMSG. □

Remark 9.3.2. *Let S' be the set of RBMSGs with a degree constraint $t \in \{1, \ldots, n\}$ at the root 0. Then the RBMSG is in S'.*

Lemma 9.3.1 and Remark 9.3.2 are basis for the choice of solution method. The solution method of the RBMSG is based on the minimum spanning tree with a single degree constraint. This is a fairly simple solution method for the problem, however it is quite efficient. Next we present a solution method for the RBMSG which we adapt from the work of Gabow [7].

9.3.1 Minimum Spanning Tree with a Single Degree Constraint

From Remark 9.3.2 it is clear that we can find the RBMSG by going through all the RBMSGs with a degree constraint at vertex 0. Moreover Lemma 9.3.1 states that RBMSG with a degree constraint t at vertex 0 is the MSTs with a degree constraint at the root 0 plus the t smallest edges from vertex $n + 1$ to vertices $\{1, \ldots, n\}$.

The problem of finding an algorithm for the minimum spanning tree (MST) problem with a single degree constraint was first discussed by Glover and Klingman in [9] where they described an algorithm with an asymptotic time of $O(|V|^2)$. In 1978 Gabow described an algorithm with an asymptotic time of $O(|E| \log \log |V| + |V| \log |V|)$ [7]. The algorithms in [9] and [7] are algorithms which iteratively go through the MSTs finding the neighboring MST with a degree constraint one higher (or one smaller) than the previous spanning tree. These algorithms could in extreme cases go through the entire set of degree constrained MSTs. An algorithm bounded by the time of generating the minimum spanning tree plus linear time was discovered by Gabow et al. [8] for the MST with a single degree constraint. So far the best asymptotic time for generating a MST is $O(|E|\alpha(|E|, |V|))$ where α is the classical functional inverse of Ackermann's function. This new

time bound was found by Chazelle [2] using a new type of heap called soft heap [3]. This means that the algorithm described in [8] by using soft heaps has the time bound of $O(|E|\alpha(|E|,|V|))$. The so far fastest algorithm for finding the MST with a single degree constraint by Gabow and Tarjan [8] does not go through all the MSTs with a single degree constraint but can in one iteration skip to a MST with a degree more than one away from the previous.

As mentioned earlier, to find the RBMSG one can go through the RBMSG with a degree constraint at the root vertex 0. The RBMSG will be the smallest of the RBMSG's with a degree constraint at vertex 0. Since the algorithm of [7], contrary to the one of [8], goes through all the single degree constrained MSTs to find the one desired one only needs to run the algorithm once to find all the single degree constraint MSTs.

In the following sections we will describe how to solve the RBMSG with an algorithm of asymptotic time $O(|E|\alpha(|E|,|V|)+|V|\log|V|)$ based on the algorithm from [7].

9.3.2 The Root Balanced Minimum Spanning Graph Algorithm

The RBMSG algorithm is based on the algorithm by Gabow [7] for finding minimum spanning trees with a single degree constraint. To find all the possible RBMSGs with degree constraints from $n-1$ to 1 we need to find all the possible MSTs with degree constraints from $n-1$ to 1 on the set $V=\{0,1,\ldots,n,n+1\}$. By fixing the desired degree constraint to 1 the algorithm by Gabow [7] will find all the MSTs starting with degree constraint 1. The remaining changes to the algorithm of [7] concern the weight of the graph and selecting the smallest of the $n-1$ RBMSGs.

Let G be a graph, let R be the edges of G to the root r and let H be an ordered list of edges from the vertices $V\setminus\{0,n+1\}$ to $n+1$ in G. Let the function $HSum(H,i)$ return the sum of the i smallest edges in H and let the function $DFS(v,T)$ return the vertices visited in a the depth first search on the tree T starting at vertex v. For two edges e and f let swapvalue(e,f) be the change in weight when removing edge e from the tree and inserting edge f. Let F(e) be a priority queue of edges f based on the swapvalue(e,f). The following pseudo code describes the algorithm for finding the RBMSG based on the algorithm for MST with a single degree constraint by Gabow [7].

RBMSG(G,r,R,H)
1: $U \leftarrow MST(G-r)$;
2: $T \leftarrow MST(R)$;
3: $X \leftarrow \emptyset$;
4: $RBMST \leftarrow T + HSum(H, degree(r))$;
5: **for** each edge $e(r,v) \in R$ **do**
6: $V' \leftarrow DFS(v, T-e)$;
7: $F(e) \leftarrow$ the edges $f(w,v) \in U$ where $w \notin V'$;
8: **if** $F(e) \neq \emptyset$ **then**
9: $f(w,v) \leftarrow$ first element in $F(e)$;
10: $X \leftarrow X \cup (e,f)$;
11: **end if**
12: **end for**
13: **while** $degree(r) > 0$ **do**
14: $e(r,v), f(v,w) \leftarrow extract_{min}(X)$;
15: $F(e) \leftarrow remove(F(e), f)$;
16: $e' \in R-e$ is the edge where $f \in F(e')$;
17: $X \leftarrow remove(X, e', f')$;
18: $F(e') \leftarrow remove(F(e'), f)$;
19: $F(e') \leftarrow F(e) \cup F(e')$;

20: **if** $F(e') \neq \emptyset$ **then**
21: $f \leftarrow min(F(e'))$;
22: $X \leftarrow insert(X, (e', f))$;
23: **end if**
24: $T \leftarrow T - e + f + HSum(H, degree(r))$;
25: **if** $T < RBMST$ **then**
26: $RBMST \leftarrow T$;
27: **end if**
28: **end while**

In line 1 the MST of the vertices in G minus the root 0 (and $n+1$) is found and stored in U and in line 2 the MST only containing edges with the root vertex 0 is found and stored in T. Then in line 3 the initial RBMST is T plus the $|T|$ smallest edges to vertex $n+1$. Note that T contains as many edges to the root 0 as possible. In lines 5 to 12 for each edge $e(r, v)$ in T the set of vertices containing vertex v spanned in $T - e$ (note that this set will not contain r) are found by a depth first search and stored in V'. All the edges $f(w, v) \neq e$ where $w \notin V'$ and $f \in U$ are inserted into a heap $F(e)$ in line 7 in the order of weight of the edge f. If there exists an swap edge f for e and consequently the set $F(e)$ is not empty then the pair of the smallest edge in $F(e)$ and e is inserted into a heap X in the order of the swap value, this is the weight of f minus the weight of e. Therefore when reaching line 13 there is at most one swap pair for each edge e in X and for each e this swap pair is the smallest. At each iteration of the while loop in lines 13 to 28 a MST with a degree constraint one less than the previous is generated. This is done by first extracting the swap pair with the smallest swap value from X (line 14) and then correct all the heaps by:

- removing the edge f from heap $F(e)$ (line 15)
- finding the edge e' connecting the root 0 to the tree containing w (line 16)
- removing the swap pair for e' in X (line 17)
- in $F(e')$ removing the edge f (line 18)
- merging $F(e)$ into $F(e')$ removing $F(e)$ (line 19)
- inserting the smallest edge f in $F(e')$ into X if such exists (lines 20 to 22)

In lines 24 to 27 the edge e is replaced by f in T reducing the degree at the root by one and the number of smallest edges connected to vertex $n+1$ corresponding to the current degree of the root is added. If this tree is smaller than the previously found degree constrained RBMST then this tree is kept as the RBMST.

The calculation of the asymptotic running time of the algorithm is described by going through the loops of the algorithm. Line 1 uses the time it takes to find the MST. The so far fastest algorithm of finding an MST by Chazelle [2] has running time $O(|E|\alpha(|E|, |V|))$ where α is the inverse Ackermann's function. This time bound also holds for line 2. Since H is sorted line 4 will at most use linear time $O(|V|)$. The sorting of H would take $O(|V| \log |V|)$ using comparison sorting. The for-loop lines 5 to 12 has running time $O(|V|)$ since there are $O(|V|)$ such edges if more than one edge exist between (w, v) then only the cheapest edge would not be superfluous.

The while loop of line 13 to 28 will be run exactly $|V| - 2$ times if there is an edge from r to all of the vertices in $\{1, \ldots, n\}$ otherwise less times depending on the maximum number of vertices in $\{1, \ldots, n\}$ which have an edge to r. Using a mergeable heap such as the Fibonacci heap developed by Fredman and Tarjan [6] for the priority queues the **extract min** operation in line 14, the **remove** operations in line 15, 17 and 18 each take $O(\log |V|)$ amortized time and the **merge** in line 19 and **insert** in line 22 each use the time $O(1)$. Finding e' in line 16 is also constant as a pointer to e' will be kept for f, note that f is at most present in two priority queues at any point

in time. The remaining operations are clearly constant. This gives an over all time bound for the while loop in lines 13 to 28 of $O(|V|\log|V|)$ and therefore time bound for the RBMSG algorithm described here is $O(|E|\alpha(|E|,|V|)+|V|\log|V|)$. Note that the asymptotic worst time for finding an RBMSG is bounded from beneath by the time it takes to find an MST as we can formulate the MST as an RBMSG problem by inserting a vertex $n+1$ in the MST problem and inserting edges of weight zero to vertex $n+1$. When removing lines 4, 25, 26 and 27 one gets the same algorithm with $b=0$ as in Gabow [7]. In Gabow [7] a proof of correctness of the algorithm is presented. Fixing b to 0 makes sure that for all possible degrees at the root r a MST with a single degree constraint is generated. The lines 4, 25, 26 and 27 updates the best RBMSG. The correctness of finding a RBMSG using MST with a single degree constraint is shown in Lemma 9.3.1 and remark 9.3.2 in Section 9.3.

9.4 The VRPTW and the RBMSGTW

As mentioned we wish to solve the VRPTW. However as we will show later the RBMSGTW is NP-hard just like the CESPPTW. To test the potential of using the RBMSGTW we will perform some preliminary tests with the VRPTW problem.

One of the properties of a good subproblem is to find solutions which are close to the solution of the original problem. In the decomposition of a VRPTW problem the subproblem will give a lower bound on the value of the original VRPTW problem. When branching this lower bound can be used to help determine and eliminate suboptimal branches in the branch and bound tree.

The test are performed on the benchmark tests constructed by Solomon [18]. The test selected are the C201 to C208, R201 to R211 and RC201 to RC208. To find the RBMSGTW we solve the RBMSG and check for feasibility on the capacity and time window constraints. If the RBMSG is not feasible a new RBMSG is found not equal to the previous and with an objective greater than or equal to the previous. Unfortunately going through the RBMSGs in order until a feasible RBMSGTW is found has been quite time consuming as often many RBMSGs are evaluated before the RBMSGTW is found. It should be noted that the number of possible spanning trees on a complete graph of n vertices is n^{n-2}.

Since the running times for the RBMSGTW are large then we have only solved for the first 7 costumers in the problem. Note that for a graph with 7 costumers there are $8^6 = 262144$ spanning trees. For the instances with 7 customers we generate the RBMSGTW and and the commonly used 2-cycle elimination shortest path. The 2-cycle elimination shortest path subproblem was introduced to the VRPTW by [5] and has since then been used in most solution algorithms to the VRPTW. This we compare to the optimal solution which is found by finding the elementary shortest path problem with resource constraints (ESPPRC).

Table 9.1 shows the lower bound generated from running the RBMSGTW compared to the lower bound generated from running the 2-cycle eliminated shortest path problem. The optimal solution generated from running the elementary shortest path problem with resource constraints is show to allow for an evaluation of the lower bounds generated. From the results in Table 9.1 it can be seen that for 8 cases RBMSGTW returns a better lower bound and for 3 cases the 2-cycle elimination shortest path return a better lowerbound. For the remaining 15 instances both the RBMSGTW and the 2-cycle eliminated shortest path return the optimal solution. It can be seen from the average of the results that the solution of the RBMSGTW on average is very close to the optimal solution whereas the average of the solutions generated by 2-cycle eliminated shortest path are significantly further away from the optimal. From these results one can conclude that by using RBMSGTW as a subproblem better lower bounds might be achieved. However, Since a valid solution to the RBMSGTW is often reached after going through a large number of RBMSGs it is not applicable in the form tested.

Lower bounds for selected VRPTW instances			
Test	RBMSGTW (LB)	2-cycle (LB)	ESPPRC (optimal solution)
7c201	1217	1217	1217
7c202	1161	1161	1161
7c203	1161	1161	1161
7c204	1161	1161	1161
7c205	1217*	1189	1217
7c206	1217*	1067	1217
7c207	1217	1217	1217
7c208	1161*	1056	1161
7r201	1888	1888	1888
7r202	1411	1411	1411
7r203	1411	1411	1411
7r204	1411	1411	1411
7r205	1647.5	1660*	1660
7r206	1411	1411	1411
7r207	1411	1411	1411
7r208	1411	1411	1411
7r209	1477.5	1507*	1538
7r210	1411	1411	1411
7r211	1411	1411	1411
7rc201	1124	1124	1124
7rc202	935*	904.5	935
7rc203	935*	904.5	935
7rc204	935*	886.6	935
7rc205	976*	935.3	976
7rc206	1016.67	1058*	1073
7rc207	976	976	976
7rc208	935*	890.2	935
Average	1246.14	1231.52	1250.93

Table 9.1: Lower bounds of RBMSGTW, 2-cycle eliminated shortest path and the optimal solution the elementary shortest path. The best lower bounds of the instance is marked with an *.

9.5 The complexity of the capacitated RBMSG with time windows

In section 9.3.2 we have shown that there exists a polynomial time algorithm for the RBMSG problem. However, here we will show that when the RBMSG is capacitated or has time windows it is in fact NP-hard. To prove the NP-completeness of the decision problems of capacitated RBMSG we will reduce from the Capacitated Tree Problem, CTP problem which is NP-complete [16].

CTP: Let $G(E, V)$ be a graph with a single source $r \in V$, an edge weight on each edge in E and a capacity c on all edges in E and let all vertices $i \in V$, $i \neq r$ have a demand d_i. Then the CTP is the problem of finding a spanning tree of overall weight less than K where on each branch from r the sum of the demands on the vertices is less than c.

D-CRBMSG: The capacitated RBMSG (CRBMSG) is the RBMSG where each edge has capacity c and the vertices 1 to n has a demand and 0 is the source. Then the D-CRBMSG is the problem of whether there exists a CRBMSG of size K or smaller.

D-CRBMSG is NP Complete: The CTP can in polynomial time be transformed to a D-CRBMSG problem by adding a vertex with 0 demand to the CTP representing the vertex $n+1$ in the D-CRBMSG problem and adding edges of cost zero from the added vertex $n+1$ to the vertices in the set $\{1, \ldots, n\}$. Clearly any solution to this D-CRBMSG would also be a solution to the CTP. Therefore the D-CRBMSG is NP-complete.

Now we will prove that the decision problem RBMSG with time windows, RBMSGTW, is NP-complete. To do this we will define the D-RBMSGTW, DCMST, D-MSTTW problems.

D-RBMSGTW: Let a graph $G(V, E)$ have a time weight t_{ij} on edge $(i, j) \in E$ and a time window (a_i, b_i) on each vertex $i \in V$. Let a vertex in V be given as the root. Given a path $P = \{r, \ldots, i-1, i\}$, let $PW(i)$ be the time weight of the path from r to i found as $PW(i) = max(PW(i-1), a_{i-1}) + t_{i-1i}$. The decision problem D-RBMSGTW is to find a RBMSG with a total time weight less than B where the time weight $PW(i)$ of a path from r to a vertex i must not exceed the ending time b_i of i's time window.

DCMST: Given a network $G = (V, E)$ with a weight $c(e)$ for each $e \in E$, delay $d(e)$ for each $e \in E$, a root $r \in V$, a delay constraint δ and a max weight B, find a spanning tree T for which the sum of the edge delays of the edges in T on the path from r to v must be less than or equal to δ and the sum of the edge weight of all the edges in T must be less than or equal to B.

D-MSTTW: Given a network $G = (V, E)$ with a weight $c(e)$ for each $e \in E$, an edge time d(e) for each $e \in E$, a time window $t(v) = (a_v, b_v)$ for each $v \in V$, a root $r \in V$, a max weight B, find a spanning tree T the sum of the edge weight of all the edges in T must be less than or equal to B and the $PW(i) \leq b_i$ for all $i \in V$.

The delay constrained minimum spanning tree DCMST was shown to be NP-complete by Salama et al. [17]. Here we show that if we can solve the D-RBMSGTW or the D-MSTTW we can also solve the DCMST.

D-MSTTW is NP complete: Any instance of the DCMST can be converted to the D-MSTTW by letting the delay be the edge time and setting each time window $t(v) = (0, \delta)$ for all $v \in V$. Since there is only one action for each edge and for each vertex then this conversion can be done in polynomial time. A solution to the generated D-MSTTW will also be a solution to the DCMST since such a solution would satisfy the requirement that the sum of the delay on the path to v for each $v \in V$ will be less than or equal to δ.

D-RBMSTTW is NP complete: Any instance of the DCMST can be converted to the D-RBMSGTW by as for the D-MSTTW case letting the delay be the edge time and setting each time window $t(v) = (0, \delta)$ for all $v \in V$. Moreover an vertex representing the $n+1$ is added to the DCMST and edges of cost zero from the added vertex $n+1$ to the vertices in the set $\{1, \ldots, n\}$ is added. Since there is only one action for each edge and two actions for each vertex then the conversion can be done in polynomial time. A solution to the generated D-RBMSGTW will also be a solution to the DCMST since such a solution would satisfy the requirement that the sum of the delay on the path to v for each $v \in V$ will be less than or equal to δ.

Moreover it can be seen that the RBMSGTW is a generalization of the bin packing problem (BPP) which is NP-hard in the strong sense. A special case of the RBMSGTW is the BPP. The case can be constructed by making the time windows non-constraining and only having cost on edges from the depot. This means that even though eliminating the problem of negative cycles the subproblem in the VRPTW problem decomposition is still hard. Even eliminating the time window requirement the remaining capacitated RBMSG decision problem D-CRBMSG is still NP-hard.

9.6 Conlusion

We have defined the RBMSG and the RBMSGTW problems which can be used in the decomposition of the VRPTW. We have developed a prototype algorithm for solving the RBMSG, based on the algorithm in [7], which can solve the RBMSG in the time $O(|E|\alpha(|E|,|V|) + |V|\log|V|)$. For dense graphs the time the $O(|E|\alpha(|E|,|V|))$ term for solving the MST would dominate the $O(|V|\log|V|)$ term and vice versa for very sparse graphs. In addition we have shown that the RBMSGTW is NP-complete.

We have described how the RBMSG can be used as a subproblem for the VRP problem and therefore also for the VRPTW problem. The preliminary tests show that using the RBMSG often generate tighter lower bounds than the 2 cycle elimination shortest path. Even though the solution times for the RBMSGTW are large for the presented prototype algorithm it might be possible to develop a competitive solution method for the RBMSGTW and thereby maybe also improve solution time of the VRPTW problem. The next step will be to implement the directed model for the RBMSGTW presented by (9.5) to (9.8) in CPLEX to see if the nice polynomial structure of the model may result in CPLEX finding a solution in reason able time.

Acknowledgments

The authors wish to thank Christian Edinger Munk Plum, Simon Spoorendonk and Brian Kallehauge for valuable comments.

Bibliography

[1] A. Chabrier. Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33:2972–2990, 2006.

[2] B. Chazelle. A minimum spanning tree algorithm with inverse- ackermann type complexity. *Journal of the ACM*, 47:1028–1047, 2000.

[3] B. Chazelle. The soft heap: An approximate priority queue with optimal error rate. *Journal of the ACM*, 47:1012–1027, 2000.

[4] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle reouting problem with time windows. *Transportation Science*, 42:387–404, 2008.

[5] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.

[6] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

[7] H. N. Gabow. A good algorithm for smallest spanning trees with a degree constraint. *Networks*, 8:201–208, 1978.

[8] H. N. Gabow and R. E. Tarjan. Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms*, 5:80–131, 1984.

[9] H. N. Glover and D. Klingman. Finding minimum spanning trees with a fixed number of links at a node. In B. Roy, editor, *Combinatorial Programming: Methods and Applications*, pages 191–200. Reidel Publishing, 1975.

[10] S. Irnich and D. Villeneuve. The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:16 pp., 2006.

[11] M. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56:497–511, 2008.

[12] B. Kallehauge. Formulations and exact algorithms for the vehicle routing problem with time windows. *Computers & Operations Research*, 35:2307–2330, 2008.

[13] B. Kallehauge, J. Larsen, and O. B. G. Madsen. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33:1464–1487, 2006.

[14] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.

[15] J. Larsen. Refinements of the column generation process for the vehicle routing problem with time windows. *Journal of Systems Science and Systems Engineering*, 13:326–341, 2004.

[16] C. H. Papadimitriou. The complexity of the capacitated tree problem. *Networks*, 8:217–230, 1978.

[17] H. F. Salama, D. S. Reeves, and Y. Viniotis. The delay-constrained minimum spanning tree problem. In *IEEE Symposium on Computers and Communications - Proceedings*, pages 699–703, 1997.

[18] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

[19] P. Toth and D. Vigo. An exact algorithm for the capacitated shortest spanning arborescence. *Annals of Operations Research*, 61:121–141, 1995.

Chapter 10

Conclusion

10.1 Summary

This thesis has covered some selected routing problems occurring in different situations such as airport passenger traffic, liner shipping, and survivable wire routing. Generally, the problems presented in this thesis consider special cases of real-life situations. As an appetizer for the problems presented in Chapter 4 to 9 a small presentation of some standard routing problems was presented in Chapter 2.

In Chapters 4 and 8 problem specific properties were investigated. In Chapters 5, 6 and 7 new real-life problems were formulated and solved. In Chapter 9 a non-standard decomposition of the vehicle routing problem with time windows were presented.

The main contributions of the thesis are:

- Development of models for new problem types:
  - Presenting a model for solving the liner shipping network design problem which includes butterfly routes and considers transshipment cost.
  - Modeling the dial-a-ride problem with synchronization points and required transfers.
  - Modeling a new generalization of the vehicle routing problem with time windows (VRPTW) where sets of edges are accessible for the route planning when a fixed cost has been paid.
- Adapting and applying solution methods to different problems:
  - Applying the branch-and-cut method to the liner shipping network design problem.
  - Applying simulated annealing with local search for the dial-a-ride problem with synchronization points and required transfers.
  - Applying the branch-cut-and-price method to the vehicle routing problem with time windows (VRPTW) containing sets of edges which are accessible for the route planning when a fixed cost has been paid.
- Realizations:
  - proving that there can not be a full Steiner tree in a solution to the 2-connected Euclidian Steiner tree problem which spans more than $\lfloor n/3 \rfloor + 1$ of the n required terminals.
  - The development of general techniques for tightening the dominance criteria when solving multi-objective non-additive shortest path problems with dynamic programming can eliminate some of the searched paths.
  - Using a new decomposition of the vehicle routing problem with time windows can be formulated. However, the subproblem is in this case NP-hard in the strong sense.

In the following the results of each chapter in Part II are summarized.

In Chapter 4 the problem of solving multi-objective non-additive shortest path problems using dynamic programming was investigated. Some general techniques for tightening the domination was developed. These techniques are developed based on real-life situations where the shortest path problem often is multi-objective and non-additive. It was in the experiments shown that the tightening developed for the domination of paths reduces the number of paths needed to be investigated when searching for the best solution.

In Chapter 5 the liner shipping problem is considered. The problem deals with the planning of a liner shipping network given a forecast of demands. The problem is modeled so that the cost of transhipment of containers at ports is included in the model along with a heterogeneous fleet and the use of butterfly routes. Moreover, the completion time of the cyclic route is used in the calculation of the capacity of a route. A branch-and-cut algorithm has been developed for solving the problem and results are reported for instances with up to 15 ports. A set of test cases [3] has been generated for testing the model and solution method. This is to the best of my knowledge the first time a branch and cut method has been applied to the liner shipping problem.

In Chapter 6 a real-life dial-a-ride problem is considered. The problem considers the transport of passengers with reduced mobility through an airport. This dial-a-ride problem application differs from other applications by the fact that cross docking is needed to transfer passengers from one mode of transport to another. The problem is modeled and a heuristic is implemented which finds promising results within minutes.

In Chapter 7 a new generalization of the VRPTW problem is presented. The generalization considers the frequently occurring constraint of paying a fixed cost for accessing a set of edges. This fixed cost could reflect payment for toll roads, investment in new facilities, the need for certifications and other costly investments. The problem has to the best of my knowledge not been formulated before. The investments considered impose a cost for the company while they also give unlimited access to a set of roads for all vehicles belonging to the company. A mathematical model of the problem is presented and solved by branch-cut-and-price, showing that the standard cuts used on the CVRP problem and the Subset Row cuts applied to the VRPTW problem also improve the solution times for this generalized version of the problem. A set of test cases [2] has been generated from the VRPTW Solomon [5] instances and problems with up to 50 customers are solved and that the branch-cut-and-price algorithm generally outperforms CPLEX.

On a more abstract level Chapter 8 considers low cost survivable networks. The structural properties of the 2-connected Euclidian Steiner network are investigated with the aim at reducing the running time for the 2-phase algorithm. The 2-phase algorithm generates full Steiner trees (FST) in the first phase and combines the generated FSTs into a Euclidean Steiner network in the second phase. The results in Chapter 8 show that only FSTs containing up to $\lfloor n/3 \rfloor + 1$ of the n required vertices needs to be generated in the first phase as a minimal 2-connected Steiner network never contains FSTs with more that $\lfloor n/3 \rfloor + 1$ required vertices.

In Chapter 9 preliminary investigations of a new decomposition of the VRPTW problem is presented. The subproblem and a preliminary solution method was presented. The subproblem was proven to be NP-hard in the strong sense. The algorithm developed for the subproblem is currently too slow when considering time windows and capacity, but surely better algorithms can and will be developed.

10.2 Perspectives and future research

In this thesis a number of new models and problem formulations were investigated. However, there are still many important routing problems occurring in real-life situations which remains to

be modeled and solved. Moreover, the models and solution methods developed here can likely be improved by further investigation. Some ideas for improvement are presented here for each article:

- In Chapter 4 multi-objective non-additive shortest path problems were discussed and general techniques for tightening the domination criteria when using dynamic programming was developed. However, there are still functions and problems where the domination criteria achieved by the techniques presented could be improved. Moreover, there are situations for which one cannot apply the general techniques, and these needs to be investigated.

- In Chapter 5 a model and a branch-and-cut solution method is presented. In the branch-and-cut method presented there are some problems with symmetry. Dealing with this symmetry problem may improve the solution time. On the modeling aspect constraining the travel time of the demands or including a delivery time window to be satisfied would definitely be useful for the shipping companies.

- In Chapter 6 a model and a heuristic for solving the problem of transporting passengers with reduced mobility through a major airport is presented. The heuristic has not been taken into use yet and it would be interesting to see how the solutions can be used in real life. Moreover, in the algorithm, the passengers are only allowed to spend time in a lounge at the terminal of departure and this restriction seems unreasonable as maybe this may put an unnecessary strain on certain parts of the route at congested times. By also allowing the passengers to wait at the arriving gate more passengers might be able to be transported as passengers with a lot of transfer time will leave space for passengers with short transfer times. Another area of investigation would be to solve the problem by constraint programming.

- In Chapter 7 the vehicle routing problem with time windows and edges set cost (ESVRPTW) was modeled and solved. The solution method used was branch-cut-and-price using Subset Row cuts. There are many recent developments in the branch-cut-and-price algorithm for the vehicle routing problem with time windows (VRPTW). Some of these improvements might with success be applied to ESVRPTW. It would definitely be interesting to see if using the methods presented in [4] and [1] on the subproblem would improve the solution time.

- In Chapter 8 a limit of $\lfloor n/3 \rfloor + 1$ on the number of required vertices in any full Steiner tree of a 2-connected Steiner network was established. Since the minimum spanning Euclidean Steiner tree has geometric properties which can help reduce the number of full Steiner trees generated it is possible that there also exists some geometric tests for the 2-connected Steiner network which can help reduce the number of full Steiner trees needed to be investigated.

- In Chapter 9 a decomposition of the VRP with the root balanced minimum spanning graph with time windows (RBMSGTW) as the subproblem is presented. A solution method was presented for the root balanced minimum spanning graph. However, a reasonable solution method for the version with time windows still has to be found. If a good solution method for RBMSGTW can be found, I believe that there might be some potential in this decomposition of the VRP problem. The next step will be to solve the subproblem with CPLEX using a directed model. This unfortunately was not tested at the time of writing this thesis.

There are endless possibilities for research in the routing and scheduling area. It is my hope that some of the results of this thesis can be helpful in connection with future research.

Bibliography

[1] R. Baldacci, E. Bartolini, A. Mingozzi, and R. Roberti. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science*, 7:229–268, 2010.

[2] L. Blander Reinhardt. http://www.diku.dk/hjemmesider/ansatte/pisinger/esvrptw.zip.

[3] L. Blander Reinhardt. http://www.diku.dk/hjemmesider/ansatte/pisinger/shipping-tests.zip.

[4] G. Desaulniers, F. Lessard, and A. Hadjar. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle reouting problem with time windows. *Transportation Science*, 42:387–404, 2008.

[5] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

In today's globalized society, transport contributes to our daily life in many different ways.The production of the parts for a shelf ready product may take place on several continents and our travel between home and work, vacation travel and business trips has increased in distance the last couple of decades.To deliver competitive service and price, transportation today needs to be cost effective.

When routing moving transportation objects such as vehicles and vessels schedules are made in connection with the routing. Such schedules represent the time for the presence of a connection between two locations. This could be an urban bus schedule where busses are routed and this routing creates a bus schedule which the passengers between locations use.

In this thesis various routing and scheduling problems will be presented. The topics covered will be routing from an origin to a destination on a predefined network, the routing and scheduling of vessels in a liner shipping network given a demand forecast to be covered, the routing of manpower and vehicles transporting disabled passengers in an airport and the vehicle routing with time windows where one version studied includes edge set cost making the cost of the individual vehicle routes interdependant.

Depending on the problem type, the size of the problems and time available for solving, different solution methods can be applicable. In this thesis both heuristic methods and several exact methods are investigated depending on the problems needed to be solved.

The solution methods applied to the problems cover dynamic programming for multi constrained shortest paths, Branch-and-cut for liner shipping, Simulated annealing for transporting assisted passengers in airports, branch-cut-and-price for vehicle routing with time windows and edges set costs.

ISBN 978-87-92706-00-3

DTU Management Engineering
Department of Management Engineering
Technical University of Denmark

Produktionstorvet
Building 424
DK-2800 Kongens Lyngby
Denmark
Tel. +45 45 25 48 00
Fax +45 45 93 34 35

www.man.dtu.dk